
Universidade de Aveiro Departamento de Engenharia Mecânica
2017

João José Redondo de Aquino

Structural numerical simulation code development with isogeometric analysis (IGA)

Desenvolvimento de códigos de simulação numérica estrutural com base em Isogeometric Analysis (IGA)

Universidade de Aveiro Departamento de Engenharia Mecânica
2017

João José Redondo de Aquino

Structural numerical simulation code development with isogeometric analysis (IGA)

Desenvolvimento de códigos de simulação numérica estrutural com base em Isogeometric Analysis (IGA)

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Mecânica, realizada sob orientação científica de Robertt Angelo Fontes Valente, Professor Associado da Universidade de Aveiro

Trabalho realizado com o apoio do Centro de Tecnologia Mecânica e Automação

o júri / the jury

presidente / president	**Prof. Doutor Rui António da Silva Moreira** Professor Auxiliar da Universidade de Aveiro
arguente	**Doutor Marco Paulo Lages Parente** Investigador Coordenador do INEGI - Instituto de Ciência e Inovação em Engenharia Mecânica e Engenharia Industrial
orientador	**Prof. Doutor Robertt Angelo Fontes Valente** Professor Associado da Universidade de Aveiro

agradecimentos / acknowledgements

Ao professor Doutor Robertt Valente por toda a motivação, pela paciência e dedicação no apoio e orientação desta última fase académica. Por me ter despertado interesse por esta área da simulação numérica e de me convencer que será o futuro na mesma.

Ao BEST Aveiro, por ter dado a possibilidade de me desenvolver como pessoa e profissional. Por trazer dos melhores momentos da minha vida académica. Por todas pessoas que tive o prazer de conhecer e de trabalhar com. Por uma lista infindável de eventos e formações que pude experienciar.

As amizades de Erasmus, com quem partilhei memórias que ficaram para sempre.

Aos amigos do curso, com quem pude contar para os trabalhos, apontamentos, estudos e não menos importante, bons momentos de lazer.

Aos amigos da Figueira da Foz pelos momentos de descontração, pelas palavras de apoio e por todos os bons momentos desde o início da minha vida académica.

À Lara, pelos momentos mais doces, pelas palavras de conforto e motivação que me deixam sempre mais relaxado, por estar sempre presente quando preciso, por tudo isto, mas podia ser só por ser quem é.

Aos meus pais, pela paciência que tiveram, pelas palavras de apoio que me dão sempre que preciso, mas principalmente pelo espírito jovem e por todas as gargalhadas de fins-de-semana bem passados, que sempre me encheram a alma.

À pessoa mais importante de todas, o meu mano, que esteve sempre ao meu lado neste percurso, do início ao fim. Por todos os momentos que partilhámos no curso, no BEST, no Erasmus, nas viagens. Somos gémeos, mas muito diferentes, tenho muito orgulho em ti. Estou eternamente agradecido pela tua companhia.

keywords

Isogeometric Analysis (IGA); Finite Element Method (FEM); Kirchhoff Classical Thin-Plate Theory; MCZ Thin plate element; Computer-Aided Engineering (CAE); Computer-Aided Design (CAD); Non-Uniform Rational B-Splines (NURBS); Rotation-Free Element

abstract

In the present day most product development industries uses the Finite Element Method (FEM) for structural analysis. Designers model the product geometries using Computer-Aided Design (CAD) software, the geometries are then fitted for analysis, by the analysts, with a mesh approximation that inevitably results in loss of accuracy. Achieving the best geometry description for complex components can be a complex task and it can take a lot of time. Considering this drawback, a new method was developed which takes advantages of curve representation tools and uses them as bases for analysis. Aiming for no loss of geometrical precision, this new method has been called "Isogeometric Analysis" (IGA).

The smoothness of Spline representations used in Isogeometric Analysis can be useful for a particular branch of structural analysis which is the analysis of plates and shells. The classic thin plate theory developed by Gustav Kirchhoff requires a geometry description with C^1 continuity between elements which is normally defined by high order polynomial functions, which typically represents a problem with the piecewise Lagrangian shape functions used in conventional FEM.

The present work explores parametric descriptions used as basis for Isogeometric Analysis, such as Bézier curves, B-splines and NURBS, taking advantage of its smoothness to develop formulations for thin plate elements. The 4-node rectangular derived by Melosh, O. Zienkiewicz and Y. Chung called MCZ thin plate element based on Kirchhoff assumptions, was the starting point to build up to a NURBS-based thin plate element.

MCZ thin plate elements, NURBS-based thin plate elements (with different order geometries) and Abaqus commercial software shell elements are evaluated by means of classical plate benchmarks comparing the elements convergences and overall performance. It can be shown that the proposed NURBS-based formulation is reliable for the analysis of thin structures.

palavras-chave

Análise Isogeométrica (IGA); Método dos Elementos Finitos (MEF); Teoria Clássica de Placas Finas de Kirchhoff ; elemento the placa MCZ; Computer-Aided Engineering (CAE); Computer-Aided Design (CAD); Non-Uniform Rational B-Splines (NURBS); Rotation-Free Element

resumo

Nos dias de hoje a maioria da indústria de desenvolvimento do produto utiliza o Método dos Elementos Finitos (MEF) na análise estrutural. Os desenhistas modelam o produto através de ferramentas de *Computer-Aided Design* (CAD). As geometrias são depois ajustadas para a análise pelos analistas que constroem uma aproximação através de uma malha de elementos finitos, o que inevitavelmente resulta numa perda de precisão geométrica. Para conseguir a melhor aproximação à geometria original para componentes complexos o processo pode ser complicado e pode consumir muito tempo. Considerando esta desvantagem foi desenvolvido um novo método que tira partido da descrição geométrica das ferramentas de desenho e utiliza as funções base das curvas para analise, com o objectivo de não haver perda de precisão geométrica, este novo método tem o nome de "Análise Isogeométrica" (IGA).

A suavidade das geometrias Splines usadas na análise isogeometrica pode ser muito útil num ramo particular da análise estrutural, no estudo das placas e cascas. A teoria clássica de análise de placas finas de Kirchhoff requer uma descrição geométrica que tenha continuidade C^1 entre elementos, que é normalmente definida por polinómios de ordem elevada, que são tipicamente um problema para as funções de forma Lagrangeanas usadas em MEF.

O presente trabalho explora as descrições geométricas utilizadas como funções de forma para a análise isogeométrica como as curvas de Bézier, as B-splines e as NURBS, tirando vantagem da facilidade de estas conseguirem a requerida continuidade entre elementos para criar elementos de placas finas com as funções de base NURBS como funções de forma. É utilizado o elemento de placa fina MCZ desenvolvido por Melosh, O. Zienkiewicz e Y. Chung com base nas premissas de Kirchhoff como ponto de partida para desenvolver o elemento com base em NURBS.

No fim os elementos de placas finas MCZ, os elementos com funções de base NURBS (com geometrias de diferentes ordens) e elementos do tipo casca do *software* comercial Abaqus são avaliados através de uma série de diferentes problemas clássicos de placas, comparando a convergência e o desempenho global. É possivel ver que a formulação proposta é fidedigna na análise de estruturas de placa fina.

Contents

Intentionally left blank.

List of Tables

Intentionally left blank.

List of Figures

List of Symbols

$\mathbf{u}$	Displacement field
ε	Strain tensor
σ	Stress field
$\mathbf{D}$	Constitutive matrix
E	Young modulus
ν	Poisson coeficient
δ	Virtual variable
$\mathbf{P}_i$	External point load vector
$\mathbf{b}$	Body force vector
$\mathbf{t}$	Surface traction vector
t	Plane thickness
N	Shape function
$\mathbf{a}$	Nodal displacement vector
$\mathbf{B}$	Bending strain matrix
$\mathbf{K}$	Stiffness Matrix
$\mathbf{f}$	Force vector
(e)	Element reference
$\mathbf{J}$	Jacobian matrix
W	Integration point weight
θ	Rotation degree of freedom
P	Position vector
B_i	Control points vector
$J_{i,n}$, $k_{j,m}$	Bernstein basis function with n, m degree

Q	Tensor product/ Surface point
$N_{i,k}$, $M_{j,l}$	kth and lth Order B-spline basis function
$R_{i,k}$	kth Order NURBS basis function
Φ	NURBS shape functions

Chapter 1

Introduction

Numerical simulation is a key technology in product development, which is spread throughout industry. Every object that we use on daily basis comes as a solution to some problem, for as simple as it might be. In industry every solution is tested before reaching market and as it gets more complex, more costs are associated with its development, being necessary to make representative models that approximate the solution to its real behavior. With the constant increasing of real-life demands, the complexity and scale of models increases as well. Model limitations are always related with processing time, computer memory or even approximation accuracy. Numerical analysis is therefore aimed to make better use of the resources for simulation with analytic and algorithmic contributions.

The dominant method in structural analysis is the Finite Element Method (mostly known as FEM), a computer based method that had its origins in the early 1960s. The designers generate a CAD (Computer Aided Design) file and these are taken into analysis, where firstly the geometry is analysis-fitted, which means that a geometric approximation is done to fit the analysis model being used. This analysis-fitted geometry is then meshed into a discrete geometry, which is an assembly of geometric elements (i.e., triangles, rectangles) that represent the whole geometry. This element mesh is then refined to make the best approximation possible to the real model. Concluding the mesh manipulation stage, model parameters are assigned and the simulation is run. With the result, concluding remarks can be satisfactory and the process is completed, otherwise the process can return to mesh manipulation in search for better results. This procedure is far from being easy, since it can take a lot of time to refining a mesh to its best approximation. It is estimated to take over than 80% of the overall analysis time in automotive, aerospace and ship building industries [1]. Another bottleneck is that the bigger the complexity of the geometry, the harder it is to make a good quality representation of the geometry approximation, which can result in loss of accuracy.

Considering the gap between designers and analysts, Thomas Hughes (2005) proposed in 2005 a new analysis concept with a tighter bond between CAD and numerical simulation. Maintaining compatibility with other traditional practices this new method is aimed towards using one, and only one, geometric model. Numerical analysis are to be performed directly on the model, this way with no loss of precision geometrically. This concept was called as Isogeometric Analysis (IGA) [1].

To do so, it was considered the vast research and bibliography [2,3] on mathematical geometry representation. The design of such free-forms shapes by mathematical methods

commenced in the 1960s. The first method to build free-forms curves and surfaces was developed by Pierre Bézier, at Renault in 1966 [4], and adopting his name. B-splines represented a further development providing more flexibility in modeling process, while Bézier curves were improving into rational-Bézier which allowed the representation of conics and circle geometries. The development of non-uniform rational B-splines (NURBS) presented the most flexible CAD technology, which permits the modeling of cylinders and spheres which are common necessary elements for representation, due to presented arguments, this technology has become standard in CAD modeling [3].

In summary, Isogeometric Analysis takes advantage of the introduced curve representations tools and uses them as basis for analysis. It has been proved that they fulfill the necessary conditions for shape functions, and therefore analysis can be performed with Bézier, B-splines or NURBS, without any geometry approximation.

An interesting characteristic of NURBS it is the curve smoothness. Using NURBS as a basis for analysis it is easy to compute a C^{p+1} curve. This is very useful for a particular branch of finite element analysis, which is the thin plate and shell analysis. A plate or a shell is a reduced or simplified model (although the real object is three-dimensional) it is described in two-dimensions, being these dimensions significantly bigger than the other one and by definition carries loads on transverse direction. The first plate theory was introduced by Gustav Kirchhoff [5] in 1850 (called the Kirchhoff thin plate theory and also known as classical plate theory). August Love [6] later developed a shell theory based on Kirchhoff assumptions and therefore, was named as Kirchhoff-Love shell theory. Solutions for classical plate theory are restricted to regular shapes, such as rectangular plates or axissimetric shells. Finite elements, already mentioned, carry a set of nodal points which are connected through what is called shape function or basis function. These functions are normally linear polynomials due to their simplicity. However, such functions typically have a C^0 continuity between elements and even higher order polynomials cannot describe a C^1 continuity between elements for arbitrary shapes. Kirchoff thin plate theory has limitations with those elements since it requires second derivatives and therefore at least C^1 continuity.

Isogeometric Analysis is a relatively new method and it is still in a "white canvas" state, where there is a lot of investigation and validation to be made. It has experienced an exponential growth over the last ten years as a lot of areas gained special interest seeking this method advantages. Shell and plate problems are one of these fields, where IGA has demonstrated to be of benefit over conventional approaches as shown in the works [7–13] . In the bending strip method - a penalty method was proposed as a solution for patch boundaries where basis are C^0 continuous in multi-patch NURBS surfaces, rotation-free IGA elements [14]. Elements with smooth boundaries such as circular or cylindrical, were successfully constructed with IGA concept [15, 16].

A 4-node rectangular thin-plate element based on Kirchhoff assumptions was derived by Melosh [17], to ensure continuity this element works under a lot of restrictions as it has to be rectangular, and its parallel sides must have the same length. These conditions brings the interest to test the behavior of said element under NURBS-basis using Isogeometric Analysis, once that with NURBS smoothness, continuity is granted. It motives to seek what advantages this analysis can deliver when compared with classical approaches, and the influence of higher continuity NURBS geometries in thin plate analysis.

This dissertation scope will focus on the development of a NURBS-based Kirchhoff thin plate element, testing the basis flexibility and comparing it with a conventional

finite element analysis trough different benchmarking problems. To reach the aim of this work a list of steps is taken into account:

a.) Implementation of the finite element method for thin plate analysis;

b.) Understand and implementation of spline geometry description used in Isogeometric Analysis;

c.) Development of NURBS-based thin plate algorithm;

d.) Application of the developed algorithms in structural benchmark problems;

e.) Results from Abaqus comercial code, using $S4$ element in the same structural benchmark problems;

f.) Results analysis and final remarks.

All the finite element analysis code was developed by the author. For the implementation of NURBS-based Kirchhoff thin plate elements, open-source toolbox *IGAFEM 2.0* was used to help facilitate some geometry descriptions processes.

The dissertation is organised as follows:

Chapter 2: Basic concepts of classical mechanic, and finite element method are introduced in order to build a background for the therms and formulation of kirchhoff plate;

Chapter 3: Kirchhoff thin plate assumptions are described, together with the formulation of the finite MCZ thin plate element, derived by Melosh [17];

Chapter 4: Spline theory is explained, from the most rudimentary mapping techniques as parametric equations, to more developed technologies as non-uniform rational B-splines (NURBS) with the goal of understanding not only, the basis but also the mesh for analysis;

Chapter 5: An Isogeometric Analysis overview is given covering only the two dimension case, also mesh refinement using non-uniform rational B-splines (NURBS) is explained;

Chapter 6: Where it is explained the integration of free-rotation NURBS-based thin plate elements.

Chapter 7: Numerical examples are presented based on common benchmark problems, NURBS-based thin plate elements will be tested and compared with other elements for plate analysis.

Chapter 8: Gives a general outlook of the resulting convergences behavior read through numerical examples. Some ideas for future works are proposed.

Intentionally left blank.

Chapter 2

Finite element method

The Finite Element Method is a numerical method that model a continuous problem as a discrete one, where from the analysis of each discrete part it is possible to obtain a mathematics description of its general behavior This chapter is a general overview of the method, as it is very important to acknowledge some crucial concepts.

In this chapter only two dimensional elements will be used to explain the finite element analysis, once these elements have more similarities to classical plate elements, that are to be explained further in the Kirchhoff thin plate theory chapter.

2.1 Plane stress and strain conditions

Kinematics deals with a continuous displacement field $\mathbf{u}$ of a point that can be defined by the two displacement components in the direction of the two coordinates x, y,

$$\mathbf{u} = \begin{bmatrix} u(x,y) \\ v(x,y) \end{bmatrix}, \tag{2.1}$$

where u and v are the displacement in each direction x and y, respectively. Strain-displacement relation can be approximated at any point of a body by the strain tensor ε_{ij}

$$\varepsilon_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right), \tag{2.2}$$

being u_i and u_j the components of the displacement field $\mathbf{u}$ of the point x_i and x_j (x, y for two dimensional bodies). Taking this into account, it is possible to obtain the strain components from the derivatives of the displacements, for a two dimensional quadrilateral element as,

$$\begin{aligned} &\varepsilon_{xx} = \frac{\partial u}{\partial x}; \quad \varepsilon_{yy} = \frac{\partial v}{\partial y}; \\ &\gamma_{xy} = 2\varepsilon_{xy} = \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}; \\ &\gamma_{xz} = \gamma_{yz} = 0. \end{aligned} \tag{2.3}$$

The strain tensor can then be written in vector form as,

$$\varepsilon = \begin{Bmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \gamma_{xy} \end{Bmatrix}. \tag{2.4}$$

There are two types of stress and strain conditions: when the strain in the z direction (perpendicular to the $x,\ y$ plane) is assigned with the zero value ($\varepsilon_{zz} = 0$) it represents a *plane strain* condition. On the other hand, when the strain differs from zero ($\varepsilon_{zz} \neq 0$) it can nevertheless generate a *plane stress* condition. In this last state, the z stress component σ_{zz} acquires the zero value ($\sigma_{zz} = 0$). The stress field is denoted as

$$\sigma = \begin{Bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \tau_{xy} \end{Bmatrix}, \tag{2.5}$$

can be related with the strain tensor trough the Hooke's law for isotropic materials in linear elasticity regime, by

$$\sigma = \mathbf{D}\varepsilon, \tag{2.6}$$

where $\mathbf{D}$ is the constitutive matrix dependent of the elastic properties of the material, given by the Young modulus, E, and also of the Poisson coefficient ν.

The Hooke's law for three dimensional deformation can then be written as

$$\begin{Bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{zz} \\ \tau_{xy} \\ \tau_{xz} \\ \tau_{yz} \end{Bmatrix} = \mathbf{D} \begin{Bmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \varepsilon_{zz} \\ \gamma_{xy} \\ \gamma_{xz} \\ \gamma_{yz} \end{Bmatrix} \quad , \tag{2.7}$$

where,

$$\mathrm{D} = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} (1-\nu) & \nu & \nu & 0 & 0 & 0 \\ \nu & (1-\nu) & \nu & 0 & 0 & 0 \\ \nu & \nu & (1-\nu) & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{(1-2\nu)}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{(1-2\nu)}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{(1-2\nu)}{2} \end{bmatrix}. \tag{2.8}$$

Calling back what is defined as the *plane stress* condition ($\sigma_{zz} = 0$), and combining the Equations 2.8 and 2.7, the strain and stress components can be related respectively by

$$\begin{cases} \varepsilon_{xx} = & \frac{\sigma_{xx}}{E} - \frac{\nu\sigma_{yy}}{E} \\ \varepsilon_{yy} = & -\frac{\nu\sigma_{xx}}{E} + \frac{\sigma_{yy}}{E} \\ \gamma_{xy} = & \frac{2(1+\nu)}{E}\tau_{xy} \end{cases}, \quad \begin{cases} \sigma_{xx} = & \frac{E}{1-\nu^2}(\varepsilon_{xx} + \nu\varepsilon_{yy}) \\ \sigma_{yy} = & \frac{E}{1-\nu^2}(\nu\varepsilon_{xx} + \varepsilon_{yy}) \\ \tau_{xy} = & \frac{E}{2(1+\nu)}\gamma_{xy} \end{cases}. \tag{2.9}$$

With this strain-stress relations, it is possible to write the definition for a *plane stress* condition in matrix form,

$$\begin{Bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \tau_{xy} \end{Bmatrix} = \frac{E}{1-\nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} \begin{Bmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \gamma_{xy}, \end{Bmatrix} \tag{2.10}$$

or as represented in Equation 2.6 here **D** is the constitutive matrix for a plane stress state. Also in a plane stress state, ε_z is defined by

$$\varepsilon_{zz} = -\frac{\nu}{E}(\sigma_{xx} + \sigma_{yy}). \tag{2.11}$$

Recalling the *plane strain* conditions $\varepsilon_{zz} = 0$ and $\sigma_{zz} \neq 0$ and applying the same procedure as was done for the *plane stress* state, in order to obtain the stress-strain relation in matrix form, the following is obtained

$$\begin{Bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \tau_{xy} \end{Bmatrix} = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-2\nu}{2} \end{bmatrix} \begin{Bmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \gamma_{xy} \end{Bmatrix}, \tag{2.12}$$

recalling the Hooke's law here, **D** is the constitutive matrix for a *plane strain* state. In this state σ_{zz} differs from zero and it is determined by

$$\sigma_{zz} = \nu(\sigma_{xx} + \sigma_{yy}). \tag{2.13}$$

2.2 Principle of Virtual Work

The principle of virtual work (PVW) is defined by a real force acting through a virtual displacement or a virtual force acting through a real displacement. For two dimensional elasticity problems it has the expression of [18],

$$\begin{aligned} \iint_A (\delta\varepsilon_{xx}\sigma_{xx} + \delta\varepsilon_{yy}\sigma_{yy} + \delta\gamma_{xy}\tau_{xy})t \ \mathrm{d}A = \iint_A (\delta u b_x + \delta v b_y)t \ \mathrm{d}A + \\ + \oint_l (\delta u t_x + \delta v t_y)t \ \mathrm{d}s + \sum_i (\delta u_i P_{x_i} + \delta v_i P_{y_i}) \end{aligned} \tag{2.14}$$

.

The terms on the left side represent the work performed by the stresses σ_x, σ_y and τ_{xy} over the virtual strains $\delta\varepsilon_x$, $\delta\varepsilon_y$ and $\delta\gamma_{xy}$. The terms on the right side represent the virtual work of body forces b_x and b_y; the surface tractions t_x and t_y and external point loads P_x, P_y, respectively. In the equation, A is the area of the solid, l is the boundary of the transverse section, t represents the thickness [18]. Equation 2.14 can be written in matrix form as

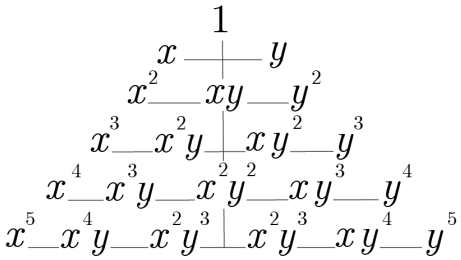
$$\iint_A \delta\varepsilon^{\mathrm{T}}\sigma t \ \mathrm{d}A = \iint_A \delta\mathrm{u}^{\mathrm{T}}\mathrm{b}t \ \mathrm{d}A + \oint \delta u^{\mathrm{T}}\mathrm{t}t \ \mathrm{d}s + \sum_i \delta\mathrm{u}_i^{\mathrm{T}}\mathrm{p}_i \tag{2.15}$$

where

$$\delta\varepsilon = \begin{Bmatrix} \delta\varepsilon_{xx} \\ \delta\varepsilon_{yy} \\ \delta\gamma_{xy} \end{Bmatrix}; \quad \delta\mathrm{u} = \begin{Bmatrix} \delta u \\ \delta v \end{Bmatrix}; \quad \mathrm{b} = \begin{Bmatrix} b_x \\ b_y \end{Bmatrix}; \quad \mathrm{t} = \begin{Bmatrix} t_x \\ t_y \end{Bmatrix}; \quad \delta\mathrm{u}_i = \begin{Bmatrix} \delta u_i \\ \delta v_i \end{Bmatrix}; \quad \mathrm{P}_i = \begin{Bmatrix} P_{x_i} \\ P_{y_i} \end{Bmatrix}.$$

2.3 Four node bilinear finite element

As announced in the chapter introduction the finite element considered for the study is a four nodes-rectangular element. As seen in the figure bellow, the polynomial that interpolates the displacement field in a rectangular element is given by the Pascal triangle where, for higher order, a higher number of nodes define a rectangle.



Polynomial Degree	Number of Terms
Constant	1
Linear	2
Quadratic	6
Cubic	10
Quartic	15
Quintic	21

Figure 2.1: Two dimensional Pascal triangle, representing the monomials for two-dimensional cases.

In a general way, and considering Figure 2.1, a finite element of order $p \geq 1$ is defined by

$$n_{nodes} = (p+1)^2. \tag{2.16}$$

Let's focus only on the 4 nodes bilinear rectangle. The approximation for the displacement field defined by the Pascal's triangle is given by,

$$u(x,y) = \alpha_1 + \alpha_2 x + \alpha_3 y + \alpha_4 xy, \tag{2.17}$$

where x, y are the global coordinates of the node being interpolated. To create the interpolation function of each displacement component of a finite element, it is required at least one shape function per node. Shape functions N are what predefine the shape of displacement variation with respect to the element coordinates [19].

Consider the following representation of a two dimensional 4-nodes rectangular element, the shape function of a node N_i, in terms of global coordinates is obtained by multiplying the unidimensional shape functions along x and y directions. As an example, the shape function of the first node N_1 is the result of the product between $N_1^{(e)}(x)$ and $N_1^{(e)}(y)$. These unidimensional shape functions are equivalent to a beam shape function. This way it is guaranteed that the shape function has the value $N_1^e(x,y) = 1$ in the interpolated node and $N_1^{(e)}(x,y) = 0$ at the other nodes.

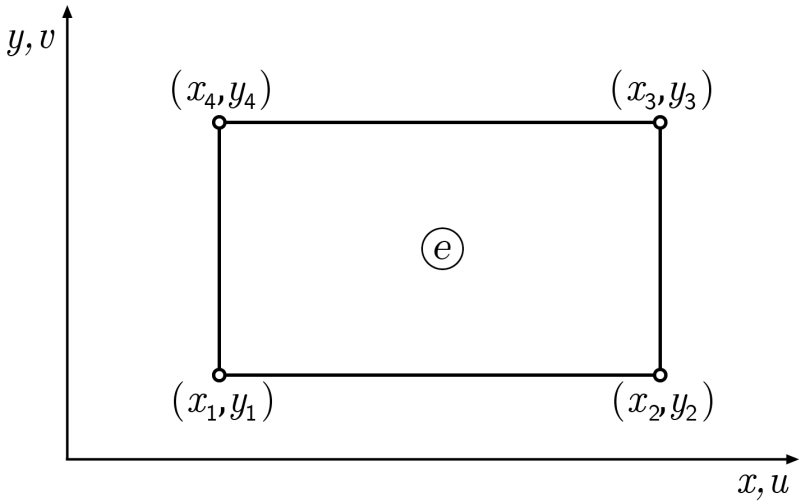


Figure 2.2: Representation of a 4-nodes Rectangular element.

The shape function of a two nodes beam element is given by,

$$N_1(x) = \left(\frac{x_2 - x}{x_2 - x_1}\right) \quad \text{and} \quad N_2(x) = \left(\frac{x - x_1}{x_2 - x_1}\right), \tag{2.18}$$

and proceeding as discussed, the shape functions of the represented rectangle element are

$$\begin{aligned}
N_1^{(e)}(x,y) =& \frac{(x-x_2)}{(x_1-x_2)}\frac{(y-y_4)}{(y_1-y_4)} = \frac{1}{A^{(e)}}(x-x_2)(y-y_4),\\
N_2^{(e)}(x,y) =& \frac{(x-x_1)}{(x_2-x_1)}\frac{(y-y_4)}{(y_1-y_4)} = \frac{1}{A^{(e)}}(x-x_1)(y-y_4),\\
N_3^{(e)}(x,y) =& \frac{(x-x_1)}{(x_2-x_1)}\frac{(y-y_1)}{(y_4-y_1)} = \frac{1}{A^{(e)}}(x-x_2)(y-y_1),\\
N_4^{(e)}(x,y) =& \frac{(x-x_2)}{(x_1-x_2)}\frac{(y-y_1)}{(y_4-y_1)} = \frac{1}{A^{(e)}}(x-x_2)(y-y_1),
\end{aligned} \tag{2.19}$$

where $A^{(e)}$ is the area of the finite element. These shape functions are extremely simple but also very limited as they can't be used to represent distorted elements. For this end, it is required to employ the isoparametric formulation as it will be seen in section 2.4.

2.3.1 Discretization of the displacement field

Considering the rectangle element shown in figure 2.2, the two cartesian displacements of a point within the element can be expressed in terms of nodal displacement, as

$$\begin{aligned}
u(x,y) =& N_1u_1(x,y) + N_2u_2(x,y) + N_3u_3(x,y) + N_4u_4(x,y),\\
v(x,y) =& N_1v_1(x,y) + N_2v_2(x,y) + N_3v_3(x,y) + N_4v_4(x,y).
\end{aligned} \tag{2.20}$$

Here $(u_i,\ v_i)$ are horizontal and vertical displacements of the node i, respectively. Bringing this equation to matrix form, it can be written as

$$\mathbf{u} = \begin{Bmatrix} u \\ v \end{Bmatrix} = \begin{bmatrix} N_1 & 0 & N_2 & 0 & N_3 & 0 & N4 & 0 \\ 0 & N_1 & 0 & N_2 & 0 & N_3 & 0 & N4 \end{bmatrix} \begin{Bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \\ u_4 \\ v_4, \end{Bmatrix} \tag{2.21}$$

or,

$$\mathbf{u} = \mathrm{N}\mathbf{a}^{(e)}, \tag{2.22}$$

where $\mathbf{u}$ is the displacement vector of a point and being $\mathbf{N}$ the matrix of nodal shape functions,

$$\mathbf{N} = [\mathrm{N}_1, \mathrm{N}_2, \mathrm{N}_3, \mathrm{N}_4] \ \ ; \ \ \mathrm{N}_i = \begin{bmatrix} N_i & 0 \\ 0 & N_i \end{bmatrix}, \tag{2.23}$$

and $\mathbf{a}^{(e)}$ the nodal displacement vector, represented as

$$\mathbf{a}^{(e)} = \begin{Bmatrix} \mathrm{a}_1^{(e)} \\ \mathrm{a}_2^{(e)} \\ \mathrm{a}_3^{(e)} \\ \mathrm{a}_4^{(e)} \end{Bmatrix} \ \text{ with } \ \mathrm{a}_i^{(e)} = \begin{Bmatrix} u_i \\ v_i \end{Bmatrix}. \tag{2.24}$$

2.3.2 Discretization of the strain and stress fields

Recalling strain vector 2.4 and substituting displacements coming from Equation 2.23, the three strains are determined in matrix form as as,

$$\varepsilon = \begin{Bmatrix} \frac{\partial u}{\partial x} \\ \\ \frac{\partial v}{\partial y} \\ \\ \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \end{Bmatrix} = \begin{bmatrix} \frac{\partial N_1}{\partial x} & 0 & \frac{\partial N_2}{\partial x} & 0 & \frac{\partial N_3}{\partial x} & 0 & \frac{\partial N_4}{\partial x} & 0 \\ \\ 0 & \frac{\partial N_1}{\partial y} & 0 & \frac{\partial N_2}{\partial y} & 0 & \frac{\partial N_3}{\partial y} & 0 & \frac{\partial N_4}{\partial y} \\ \\ \frac{\partial N_1}{\partial y} & \frac{\partial N_1}{\partial x} & \frac{\partial N_2}{\partial y} & \frac{\partial N_2}{\partial x} & \frac{\partial N_3}{\partial y} & \frac{\partial N_3}{\partial x} & \frac{\partial N_4}{\partial y} & \frac{\partial N_4}{\partial x} \end{bmatrix} \begin{Bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \\ u_4 \\ v_4 \end{Bmatrix}, \tag{2.25}$$

or

$$\varepsilon = \mathbf{B}\mathbf{a}^{(e)}, \tag{2.26}$$

where $\mathbf{B}$ is the element strain matrix

$$\mathbf{B} = [\mathbf{B}_1, \mathbf{B}_2, \mathbf{B}_3, \mathrm{B}_4] \ \ ; \ \text{with} \ \ \mathbf{B}_i = \begin{bmatrix} \frac{\partial N_i}{\partial x} & 0 \\ 0 & \frac{\partial N_i}{\partial y} \\ \frac{\partial N_i}{\partial y} & \frac{\partial N_i}{\partial x} \end{bmatrix}, \tag{2.27}$$

being B_i the strain matrix of node i.

The discretized expression for the element stress field is obtained by replacing Equation 2.26 into 2.6, as

$$\sigma = \mathbf{D}\varepsilon = \mathbf{BDa}^{(e)}. \tag{2.28}$$

2.3.3 Discretization of the equilibrium equations

The discretization of the equilibrium equations of a four node rectangular element will be determined by applying the principle of virtual work to an element. Let's assume that the following forces act on the element: **b**, body forces, distributed forces acting per unit area; and **t**, surface tractions, distributed forces along side of the element boundary line.

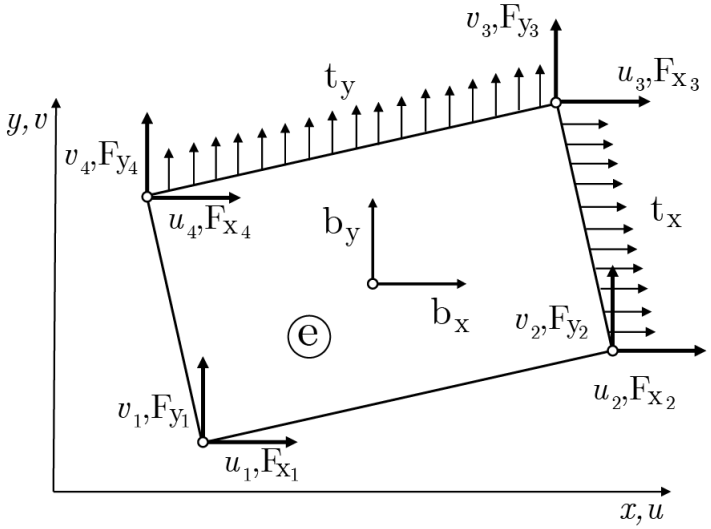


Figure 2.3: Forces acting in a 4-noded rectangle: sides 2-3 and 3-4 belong to external boundary.

As it is usual in this type of analysis, the equilibrium of forces acting on the element is imposed point-wise at the nodes. It is required to define the nodal point loads F_x and F_y in order to balance internal and external forces to the element deformation. This balance is obtained by applying the PVW,

$$\begin{aligned}\iint_{A^{(e)}} \delta\varepsilon^{\mathrm{T}}\sigma t \, \mathrm{d}A = \iint_{A^{(e)}} \delta\mathbf{u}^{\mathrm{T}}\mathbf{b}t \, \mathrm{d}A + \oint_{l^{(e)}} \delta\mathbf{u}^{\mathrm{T}}\mathbf{t}t \, \mathrm{d}s + \\ + \sum_{i=1}^{4} \delta u_i F_{x_i} + \sum_{i=1}^{4} \delta v_i F_{y_i},\end{aligned} \tag{2.29}$$

where δu_i and δv_i are the nodal virtual displacements and F_{x_i} and F_{y_i} are the balancing nodal forces along horizontal and vertical directions, respectively. The virtual work done

by these forces is,

$$\iint_{A^{(e)}} \delta\varepsilon^{\mathrm{T}}\sigma t \ \mathrm{d}A - \iint_{A^{(e)}} \delta\mathbf{u}^{\mathrm{T}}\mathbf{b}t \ \mathrm{d}A - \oint_{l^{(e)}} \delta\mathbf{u}^{\mathrm{T}}\mathbf{t}t \ \mathrm{d}s = [\delta\mathbf{a}^{(e)}]^{\mathrm{T}}\mathrm{q}^{(e)}, \tag{2.30}$$

where, for a 4-noded rectangular element,

$$\delta a^{(e)} = \begin{bmatrix} \delta a_1 \\ \vdots \\ \delta a_4 \end{bmatrix}^{(e)} = \begin{bmatrix} \delta u_1, \delta v_1 \\ \vdots \\ \delta u_4, \delta v_4 \end{bmatrix}^{(e)} \quad q^{(e)} = \begin{bmatrix} q_1 \\ \vdots \\ q_4 \end{bmatrix}^{(e)} = \begin{bmatrix} F_{x1}, F_{y1} \\ \vdots \\ F_{x4}, F_{y4} \end{bmatrix}$$

Recalling from the last section, the strain and stress Equations 2.22 and 2.26 and following the same procedures,

$$\delta\mathrm{u} = \mathbf{N}\delta\mathbf{a}^{(e)} \ \ ; \ \ \delta\varepsilon = \mathbf{B}\delta\mathrm{a}^{(e)} \tag{2.31}$$

replacing these equations into the Equation 2.30 results as

$$[\delta\mathbf{a}^{(e)}]^{\mathrm{T}}\left[\iint_{A^{(e)}} \mathbf{B}^{\mathrm{T}}\sigma t \ \mathrm{d}A - \iint_{A^{(e)}} \mathrm{N}^{\mathrm{T}}\mathbf{b}t \ \mathrm{d}A - \oint_{l^{(e)}} \mathbf{N}^{\mathrm{T}}\mathbf{t}t \ \mathrm{d}s\right] = q^{(e)}[\delta\mathbf{a}^{(e)}]^{\mathrm{T}}. \tag{2.32}$$

Given the arbitrary of the virtual displacements, the previous equation can be rewritten as

$$\iint_{A^{(e)}} \mathbf{B}^{\mathrm{T}}\sigma t \ \mathrm{d}A - \iint_{A^{(e)}} \mathbf{N}^{\mathrm{T}}\mathbf{b}t \ \mathrm{d}A - \oint_{l^{(e)}} \mathbf{N}^{\mathrm{T}}\mathbf{t}t \ \mathrm{d}s = q^{(e)} \tag{2.33}$$

The Equation 2.33 balances the nodal forces $q^{(e)}$ due to element deformation, body forces and surface tractions. Substituting the stresses in terms of the nodal displacements from Equation 2.28 gives

$$\iint_{A^{(e)}} \mathbf{B}^{\mathrm{T}}(\mathbf{DBa}^{(e)} - \mathbf{D}\varepsilon^0 + \sigma^0)t \ \mathrm{d}A - \iint_{A^{(e)}} \mathbf{N}^{\mathrm{T}}\mathbf{b}t \ \mathrm{d}A - \oint_{l^{(e)}} \mathbf{N}^{\mathrm{T}}\mathbf{t}t \ \mathrm{d}s = q^{(e)}. \tag{2.34}$$

In matrix form, this can be expressed as,

$$\mathbf{K}^{(e)}\mathbf{a}^{(e)} - \mathbf{f}^{(e)} = \mathbf{q}^{(e)}, \tag{2.35}$$

where $\mathbf{K}$ is the stiffness matrix, obtained by

$$\mathbf{K}^{(e)} = \iint_{A^{(e)}} \mathbf{B}^{\mathrm{T}}\mathbf{DB}t \ \mathrm{d}A, \tag{2.36}$$

where,

$$\mathbf{f}^{(e)} = \mathbf{f}_{\varepsilon}^{(e)} + \mathbf{f}_{\sigma}^{(e)} + \mathbf{f}_{b}^{(e)} + \mathbf{f}_{t}^{(e)} \tag{2.37}$$

is the equivalent nodal force vector for the element, also written by

$$\mathbf{f}^{(e)} = \iint_{A^{(e)}} \mathbf{B}^{\mathrm{T}}\mathbf{D}\varepsilon^0 t \ \mathrm{d}A - \iint_{A^{(e)}} \mathbf{B}^{\mathrm{T}}\sigma^0 t \ \mathrm{d}A + \iint_{A^{(e)}} \mathbf{N}^{\mathrm{T}}\mathbf{b}t \ \mathrm{d}A + \oint_{l^{(e)}} \mathbf{N}^{\mathrm{T}}\mathbf{t}t \ \mathrm{d}s. \tag{2.38}$$

In this equation, are shown the equivalent nodal force vectors due to initial strains, initial stresses, body forces and surface tractions, respectively.

The global equilibrium equations for the whole mesh are obtained by establishing that the nodes are in equilibrium. Being $\mathbf{p}_j = [P_{x_j}, P_{y_j}]$ the reactions to the sum of all load acting at the node j

$$\sum_e \mathbf{q}_i^{(e)} = \mathbf{p}_j \ , \quad j = 1, n_{nodes}, \tag{2.39}$$

here, the sum refers to all elements that share the same node j and n_{nodes} the number of nodes.

2.4 Isoparametric concept

As was mentioned in the previous section, the shape functions shown have a drawback when interpolating in elements with curved boundaries. This can be overcome through the isoparametric concept and numerical integration, respectively. The isoparametric interpolation in two dimension in terms of natural coordinates ξ, η, is based in the following parametrization for the position (x, y) of any point inside any given element,

$$\begin{cases} x = \sum_{i=1}^{n_{nodes}} N_i(\xi, \eta) x_i \\ \\ y = \sum_{i=1}^{n_{nodes}} N_i(\xi, \eta) y_i \end{cases} , \tag{2.40}$$

where n_{nodes} is the number of nodes of the finite element. In the natural referential system (ξ, η), the coordinates of any point of the element are defined in $[-1, +1] \times [-1, +1]$, independently of the size and distortion of its boundaries, as represented in Figure 2.4. The shape functions are based on corresponding Lagrange polynomials defined on the

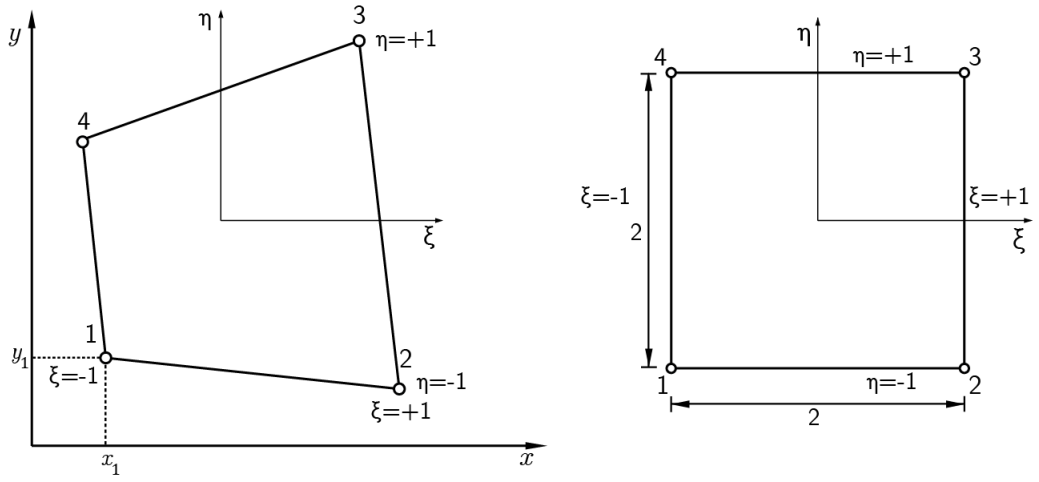


Figure 2.4: Representation of the actual and normalized geometry of a four-noded isoparamteric quadrilateral.

natural coordinate system (ξ, η). For global coordinates (x, y), these functions are obtained from a combination of unidimensional shape functions. The unidimensional shape

function for the natural coordinate ξ can be determined by the generic formulation of Lagrange unidimensional polynomials,

$$N_i(\xi) = \prod_{j=1(j\neq i)}^{n_{nodes}} \frac{(\xi - \xi_j)}{(\xi_i - \xi_j)}, \tag{2.41}$$

where, by definition, $\xi \in [-1, +1]$. Analogously for the natural coordinate η,

$$N_i(\eta) = \prod_{j=1(j\neq i)}^{n_{nodes}} \frac{(\eta - \eta_j)}{(\eta_i - \eta_j)}, \tag{2.42}$$

with $\eta \in [-1, +1]$.

For the chosen four nodes rectangular element, the shape functions of each node are

$$\begin{cases} N_1 = \frac{1}{4}(1-\xi)(1-\eta) \\ N_2 = \frac{1}{4}(1+\xi)(1-\eta) \\ N_3 = \frac{1}{4}(1+\xi)(1+\eta) \\ N_4 = \frac{1}{4}(1-\xi)(1+\eta) \end{cases} . \tag{2.43}$$

The shape functions for this element can then be represented generically as

$$N_i = \frac{1}{4}(1 + \xi\xi_i)(1 + \eta\eta_i) \ , \tag{2.44}$$

where ξ_i and η_i represents the natural coordinates of the i node. Note that Equation 2.44 follows the essential conditions of the shape functions, that are

$$N_i(\xi_j, \eta_j) = \begin{cases} 1 & \text{if } \ i = j \\ 0 & \text{if } \ i \neq j \end{cases} \ , \tag{2.45}$$

Figure 2.5 summaries how to determine the shape functions for a 4-noded rectangular element in two dimensions.

In order to determine the strain field it is necessary to calculate the variations of the shape functions relatively to the infinitesimal variations of the position in the global coordinate system. This derivatives were already introduced in the Equation 2.2. Let's reintroduce the matrix that relates the displacement field with the strain field,

$$\mathbf{B} = [\mathbf{B}_1, \mathbf{B}_2, \mathbf{B}_3, \mathbf{B}_4] \ \ ; \ \text{with} \ \ \mathbf{B}_i = \begin{bmatrix} \frac{\partial N_i}{\partial x} & 0 \\ 0 & \frac{\partial N_i}{\partial y} \\ \frac{\partial N_i}{\partial y} & \frac{\partial N_i}{\partial y} \end{bmatrix},$$

To build such matrices a coordinate transformation is needed from the natural space to the global coordinate system, and the use of the chain rule is required

$$\begin{aligned} \frac{\partial N_i(\xi,\eta)}{\partial \xi} &= \frac{\partial N_i(\xi,\eta)}{\partial x}\frac{\partial x}{\partial \xi} + \frac{\partial N_i(\xi,\eta)}{\partial y}\frac{\partial y}{\partial \xi}, \\ \frac{\partial N_i(\xi,\eta)}{\partial \eta} &= \frac{\partial N_i(\xi,\eta)}{\partial x}\frac{\partial x}{\partial \eta} + \frac{\partial N_i(\xi,\eta)}{\partial y}\frac{\partial y}{\partial \eta}, \end{aligned} \tag{2.46}$$

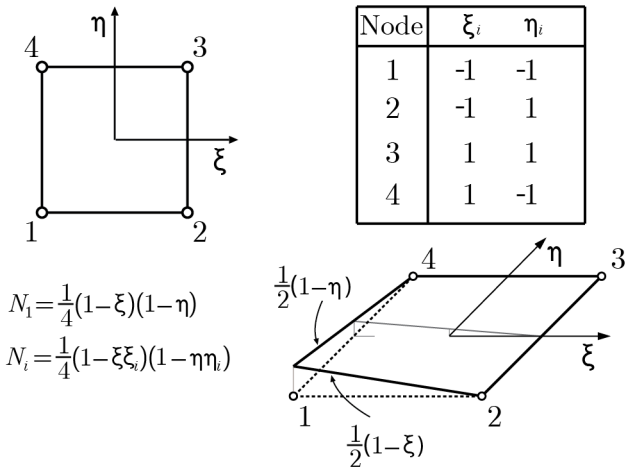


Figure 2.5: 4-noded Lagrangian rectangular element.

or, presenting in a matrix form,

$$\begin{Bmatrix} \frac{\partial N_i(\xi,\eta)}{\partial \xi} \\ \frac{\partial N_i(\xi,\eta)}{\partial \eta} \end{Bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix} \begin{Bmatrix} \frac{\partial N_i(\xi,\eta)}{\partial x} \\ \frac{\partial N_i(\xi,\eta)}{\partial y} \end{Bmatrix}. \tag{2.47}$$

From the previous expression, the Jacobian matrix, can be defined in the form

$$\mathbf{J} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix}. \tag{2.48}$$

This matrix is responsible for the mapping between both coordinate systems, connecting the differentials of (x, y) to (ξ, η) and vice-versa. With the inverse of the Jacobian matrix, assuming that exists, it can be determined that,

$$\begin{Bmatrix} \frac{\partial N_i(\xi,\eta)}{\partial x} \\ \frac{\partial N_i(\xi,\eta)}{\partial y} \end{Bmatrix} = \mathbf{J}^{-1} \begin{Bmatrix} \frac{\partial N_i(\xi,\eta)}{\partial \xi} \\ \frac{\partial N_i(\xi,\eta)}{\partial \eta} \end{Bmatrix}. \tag{2.49}$$

Finally, the Jacobian matrix determinant relates the the partial derivatives from the cartesian referential with the derivatives from isoperimetric one, by the following way,

$$\mathrm{d}x\mathrm{d}y = |\mathbf{J}|\ \mathrm{d}\xi\mathrm{d}\eta. \tag{2.50}$$

2.5 Numerical integration

All element integrals can be written in natural coordinate space making use of isoparametric formulation. The numerical integration by Gauss quadrature will be considered [18].

The integral of any general term $g(\xi, \eta)$ over normalized isoparametric quadrilateral domain can be determined by using Gauss quadrature as,

$$\int_{-1}^{+1}\int_{-1}^{+1} g(\xi,\eta)\ \mathrm{d}\xi\mathrm{d}\eta = \int_{-1}^{+1}\left[\sum_{q=1}^{n_q} g(\xi,\eta_q)W_q\right]\mathrm{d}\xi = \sum_{p=1}^{n_p}\sum_{q=1}^{n_q} g(\xi_p,\eta_q)W_pW_q \qquad (2.51)$$

being n_q and n_p the number of integration points along each coordinate ξ and η, respectively, ξ_p and η_p are the natural coordinates of the pth integration point, W_p and W_q are the corresponding weights from each integration point. The coordinates of the integration points result from the integration rules of Gauss-Legendre [20].

Chapter 3

Thin plates theory

A plate structure is geometrically similar to a bi-dimensional stress problem, whereas in this case it carries loads in the perpendicular plane, this loads lead to the bending of the structure. Plates can be described as structures where thickness is significantly smaller than other dimensions [19]. The approach on the deformation of plates is done by approximations from a three-dimensional space into a two-dimensional problem. This approach shows advantages over thin solid elements, such as the reduction of degrees of freedom as the potential for improved accuracy through the elimination of troublesome terms. The classic thin plate theory was developed by Kirchhoff [5] and is often associated with his name. Kirchhoff theory establishes that the normal remains straight and orthogonal to the middle plane after deformation, this last assumption is what differs from the more advanced thick plate theory formalized by Reissner [21] and Mindlin [22], which establishes that the normal remains straight but not necessarily orthogonal to the middle plate. Notwithstanding, on this work, only Kirchhoff theory will be adopted, hence the interest of this study falls in the required C^1 continuity to be discussed in further chapters.

In this chapter it will be presented the Kirchhoff plate theory formulation. "Non-conforming" elements formulation will also be introduced, these are elements where the rotations are discontinuous along the common element boundaries and can satisfy C^1 continuity.

3.1 Thin plate formulation

As introduced before a plate is a structure with the thickness significantly smaller than the two other dimensions whereas the middle plane is equidistant from the upper and lower surface, and it is taken as reference plane ($z = 0$) for deriving the kinematic equations. The classic plate model is based on some kinematic assumptions made by Krichhoff for its derivation: First assumption implies that every point that shares the same transverse normal to the middle plane has the same vertical displacement;

$$w(x, y, z) = w(x, y) \tag{3.1}$$

The second assumption tells that the transverse normal of the middle plane remains straight during the deformation which implies that, the in-plane displacement field is

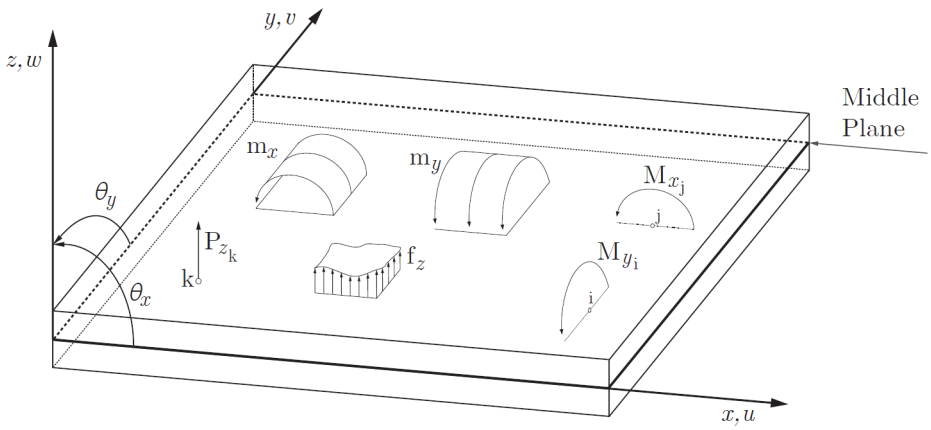


Figure 3.1: Geometric definition of a plate: sign convention for loads, moments, displacements and rotations (adapted from [23]).

almost linear in terms to z coordinate;

$$u(x, y, z) = -z\theta_x(x, y) \tag{3.2}$$

$$v(x, y, z) = -z\theta_y(x, y). \tag{3.3}$$

In the sketch presented by the Figure 3.1 the meaning of in-plane rigid rotations θ of the normal to the middle plane, is represented. Note that, by convention, both rotations have a positive value, the negative sign in the equations 3.2 and 3.3 will create a negative displacement along x and y directions;

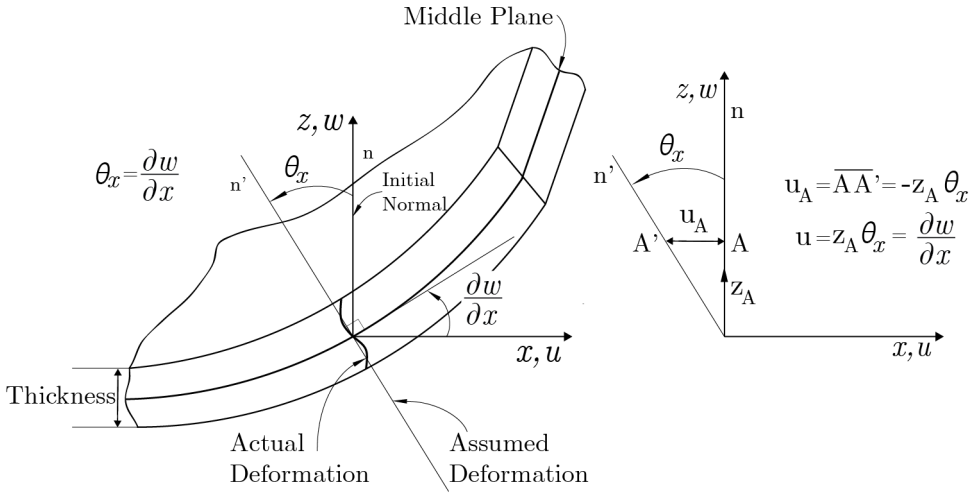


Figure 3.2: Representation of in plane displacement (adapted from [23]).

The third assumption state that a straight line normal to the undeformed plate, remains

normal to the deformed middle plane (normal orthogonality condition), this implies that rotation of the normal can be obtained from the derivatives of the reference surface as represented in the Figure 3.2, being defined by

$$\theta_x = \frac{\partial w}{\partial x}; \quad \theta_y = \frac{\partial w}{\partial y}. \tag{3.4}$$

It is possible to change the variables θ_x and θ_y in the Equations 3.2 and 3.3,getting the Krichhoff displacement field for a plate structure as

$$\begin{aligned} u(x,y,z) &= -z\frac{\partial w(x,y)}{\partial x}, \\ v(x,y,z) &= -z\frac{\partial w(x,y)}{\partial y}, \\ w(x,y,z) &= w(x,y), \end{aligned} \tag{3.5}$$

and the displacement vector written as

$$\mathbf{u} = \left[w, \frac{\partial w}{\partial x}, \frac{\partial w}{\partial y}\right]. \tag{3.6}$$

3.2 Stress and strain fields

Considering the strain-displacement relations and the Kirchhoff displacement field for plate structures, it is possible to define the strain field of the thin plate,

$$\begin{aligned} \varepsilon_{xx} &= \frac{\partial u}{\partial x} = -z\frac{\partial^2 w}{\partial x^2}, \\ \varepsilon_{yy} &= \frac{\partial v}{\partial y} = -z\frac{\partial^2 w}{\partial y^2}, \\ \varepsilon_{zz} &= 0, \\ \gamma_{xy} &= \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} = -2z\frac{\partial^2 w}{\partial x \partial y}, \\ \gamma_{xz} &= \frac{\partial w}{\partial x} + \frac{\partial u}{\partial x} = \frac{\partial w}{\partial x} - \frac{\partial w}{\partial x} = 0, \\ \gamma_{yz} &= \frac{\partial w}{\partial y} + \frac{\partial v}{\partial y} = \frac{\partial w}{\partial y} - \frac{\partial w}{\partial y} = 0. \end{aligned} \tag{3.7}$$

The value of ε_{zz} is a direct consequence of the plane stress assumption which implies that the normal stress σ_{zz} is negligible. It is also represented, in the Equation 3.7, the

normal orthogonality assumption that leads to zero transverse shear strains, meaning that these shear strains do not contribute to the deformation work. Note that this does not mean that these stresses are insignificant.

The strain vector is defined by the three significant strains,

$$\varepsilon = [\varepsilon_{xx}, \varepsilon_{yy}, \gamma_{xy}]^{\mathrm{T}} = \left[-z\frac{\partial^2 w}{\partial x^2}, -z\frac{\partial^2 w}{\partial y^2}, -2z\frac{\partial^2 w}{\partial x \partial y}\right] = \mathbf{S}\hat{\varepsilon}_b, \tag{3.8}$$

being

$$\mathbf{S} = -z\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix} = -z\mathbf{I}_3, \quad \hat{\varepsilon}_b = \left[\frac{\partial^2 w}{\partial x^2}, \frac{\partial^2 w}{\partial y^2}, \frac{\partial^2 w}{\partial x \partial y}\right], \tag{3.9}$$

where $\hat{\varepsilon}_b$ is the generalized strain vector. Matrix $\mathbf{S}$ transforms the curvatures of the middle plane into strains.

Once known the strain vector, the stress vector is easily obtained through Hooke's 3D isotropic material law Equation 2.10 and the constitutive matrix $\mathbf{D}$ defined according to the constitutive law for *plane stress* conditions equation.

$$\mathbf{D} = \frac{E}{(1-\nu^2)}\begin{bmatrix}1 & \nu & 0\\ \nu & 1 & 0\\ 0 & 0 & \frac{1-\nu}{2}\end{bmatrix}. \tag{3.10}$$

3.3 Bending moments and shear forces

It is necessary to integrate the several stresses through the plate thickness to evaluate the resultant forces and moments acting. This stress vector, known as bending moment vector, is defined as follows:

$$\hat{\sigma}_b = \begin{Bmatrix}M_x\\ M_y\\ M_{x,y}\end{Bmatrix} = \int_{-\frac{t}{2}}^{+\frac{t}{2}} \mathbf{S}^{\mathrm{T}}\begin{Bmatrix}\sigma_x\\ \sigma_y\\ \gamma_{xy}\end{Bmatrix}\mathrm{d}z = \int_{-\frac{t}{2}}^{+\frac{t}{2}} \mathbf{S}^{\mathrm{T}}\sigma\ \mathrm{d}z, \tag{3.11}$$

where the stress σ_x and σ_y lead to the bending moments M_x and M_y, respectively and the tangential stress γ_{xy} produces torque M_{xy}. Also being S a diagonal matrix implies that $\mathbf{S}^T \equiv \mathbf{S}$. The sign convention for moments M_x and M_y is consistent with the rotations θ_x and θ_y, respectively.

Recalling the Hooke's law and substituting into the Equation 3.11 yields,

$$\hat{\sigma}_b = \int_{-\frac{t}{2}}^{+\frac{t}{2}} \mathbf{S}^{\mathrm{T}}\mathbf{D}\varepsilon\ \mathrm{d}z = \int_{-\frac{t}{2}}^{+\frac{t}{2}} \mathbf{S}^{\mathrm{T}}\mathbf{D}\mathbf{S}\hat{\varepsilon}_b\ \mathrm{d}z = \hat{\mathbf{D}}_b\hat{\varepsilon}_b, \tag{3.12}$$

with the generalized constitutive matrix $\hat{\mathbf{D}}_b$ being defined as

$$\hat{\mathbf{D}}_b = \frac{t^3}{12}\mathbf{D}. \tag{3.13}$$

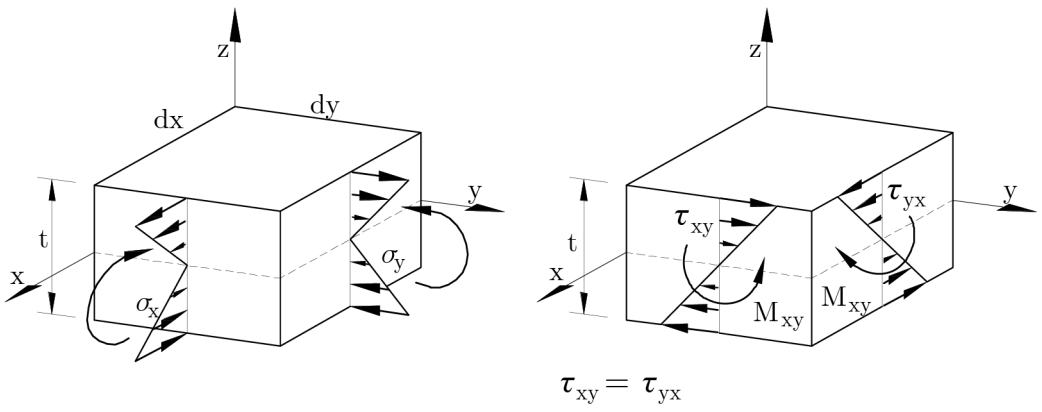


Figure 3.3: Representation of in sign convention for bending stresses and moments.

3.4 Principle of Virtual Work for a thin plate

The principle of virtual work for a thin plate is written as,

$$\iiint_V \delta\varepsilon^{\mathrm{T}}\sigma \, \mathrm{d}V = \iint_A \delta\mathbf{u}^{\mathrm{T}}\mathbf{t} \, \mathrm{d}A + \sum_i \delta\mathbf{u}_i^{\mathrm{T}}\mathbf{p}, \tag{3.14}$$

where

$$\delta\mathbf{u} = \left[\delta w, \delta\left(\frac{\partial w}{\partial x}\right), \delta\left(\frac{\partial w}{\partial y}\right)\right]^{\mathrm{T}} \quad , \quad \delta\mathbf{u}_i = \left[\delta w_i, \delta\left(\frac{\partial w}{\partial x}\right)_i, \delta\left(\frac{\partial w}{\partial y}\right)_i\right]^{\mathrm{T}},$$
$$\mathbf{t} = [f_z, m_x, m_y]^{\mathrm{T}} \quad , \quad \mathbf{p}_i = [P_{z_i}, M_{x_i}, M_{y_i}]^{\mathrm{T}}.$$

In the previous equations $\delta\mathbf{u}$ is defined as the virtual displacement, $\mathbf{t}$ is the distributed force vector, being its components f_z a distributed vertical force vector, m_x and m_y are the bending moments around the x and y axis, respectively, $\mathbf{p}_i$ is the point force vector being F_{x_i}, M_{x_i} and M_{y_i} its components relating the external vertical point load and the bending moments acting at point i, respectively.

From the Equations 3.8 and 3.11 it is possible to simplify the virtual displacement, transforming the volume integral to a surface one, replacing the referred equations into Equation 3.11, results in,

$$\begin{aligned}\delta U &= \iiint_V \delta\varepsilon^{\mathrm{T}}\sigma \, \mathrm{d}V = \iiint_V [\mathbf{S}\delta\hat{\varepsilon}_b^{\mathrm{T}}]^{\mathrm{T}}\sigma \, \mathrm{d}V = \\ &= \iint_A \delta\hat{\varepsilon}_b^{\mathrm{T}} \left[\int_{-\frac{t}{2}}^{+\frac{t}{2}} \mathbf{S}^T\sigma \, \mathrm{d}z\right] \mathrm{d}A = \iint_A \delta\hat{\varepsilon}_b^{\mathrm{T}}\hat{\sigma}_b \, \mathrm{d}A.\end{aligned} \tag{3.15}$$

Substituting into the PVW equation 3.11, it is rewritten as,

$$\iint_A \delta\hat{\varepsilon}_b^{\mathrm{T}}\sigma_b \, \mathrm{d}A = \iint_A \delta\mathbf{u}^{\mathrm{T}}\mathbf{t} \, \mathrm{d}A + \sum_i \delta\mathbf{u}_i^{\mathrm{T}}\mathbf{p}_i. \tag{3.16}$$

All variable in the PVW are functions of the coordinates of the middle plane, meaning that a plate can be treated as a two-dimensional solid for analysis.

Rewriting the virtual strain equation it is possible to clearly understand that C^1 continuity is needed for the deflection field, as it contains second derivatives of it.

$$\delta U = \iint_A \left[\frac{\partial^2 w}{\partial x^2} M_x + \frac{\partial^2 w}{\partial y^2} M_y + 2 \frac{\partial^2 w}{\partial x \partial y} M_{xy} \right] \mathrm{d}A. \tag{3.17}$$

3.5 Equilibrium equations

The equilibrium equations allow the computation of the shear forces through nodal displacement. The equilibrium of external forces, bending moments and shear forces over a differential plate element under distributed vertical loads is,

$$\sum F_z = 0 \Rightarrow \left(\frac{\partial Q_x}{\partial x} \mathrm{d}x \right) \mathrm{d}y + \left(\frac{\partial Q_y}{\partial y} \mathrm{d}y \right) \mathrm{d}x + f_z \mathrm{d}x \mathrm{d}y = 0, \tag{3.18}$$

and dividing the Equation 3.18 by its differential area($\mathrm{d}x\mathrm{d}y$) yields,

$$\left(\frac{\partial Q_x}{\partial x} \right) + \left(\frac{\partial Q_y}{\partial y} \right) + f_z = 0. \tag{3.19}$$

Equilibrium also implies that the summation of moments should be zero, that is

$$\begin{aligned} \sum M_x = 0 \Rightarrow & \left(\frac{\partial M_x}{\partial x} \mathrm{d}x \right) \mathrm{d}y + \left(\frac{\partial M_{xy}}{\partial y} \mathrm{d}y \right) \mathrm{d}x + \\ & + (Q_x \mathrm{d}y) \mathrm{d}x + \left(\frac{\partial Q_y}{\partial y} \mathrm{d}y \right) \mathrm{d}x \frac{\mathrm{d}x}{2} - f_z \mathrm{d}x \mathrm{d}y \frac{\mathrm{d}y}{2}, \\ \sum M_y = 0 \Rightarrow & \left(\frac{\partial M_y}{\partial y} \mathrm{d}y \right) \mathrm{d}x + \left(\frac{\partial M_{xy}}{\partial x} \mathrm{d}x \right) \mathrm{d}y + \\ & + (Q_y \mathrm{d}x) \mathrm{d}y + \left(\frac{\partial Q_x}{\partial x} \mathrm{d}x \right) \mathrm{d}y \frac{\mathrm{d}y}{2} - f_z \mathrm{d}x \mathrm{d}y \frac{\mathrm{d}x}{2}. \end{aligned} \tag{3.20}$$

By ignoring the second order terms (the last two terms of each equation), it follows

$$\left(\frac{\partial M_x}{\partial x} \right) + \left(\frac{\partial M_{xy}}{\partial y} \right) + Q_x = 0, \tag{3.21}$$

$$\left(\frac{\partial M_y}{\partial y} \right) + \left(\frac{\partial M_{xy}}{\partial x} \right) + Q_y = 0. \tag{3.22}$$

Differentiating 3.21 with respect to x and Equation 3.22 to y and substituting in the vertical force equilibrium Equation 3.19 it gives,

$$\frac{\partial^2 M_x}{\partial x^2} + 2 \frac{\partial^2 M_{xy}}{\partial x \partial y} + \frac{\partial^2 M_y}{\partial y^2} - f_z = 0. \tag{3.23}$$

Rewritten for an isotropic material, it comes that

$$\frac{\partial^4 w}{\partial x^4} + 2 \frac{\partial^4 w}{\partial x^2 \partial y^2} + \frac{\partial^4 w}{\partial y^4} - \frac{f_z}{D} = 0, \tag{3.24}$$

being,

$$D = \frac{Et^3}{12(1-\nu^2)} \tag{3.25}$$

The fourth order differential Equation 3.24 relates the deflection to the applied distributed load and the material properties of the plate. Substituting 3.19 for the simplified Equations 3.21 and 3.22, comes that

$$Q_x = -D\left(\frac{\partial^3 w}{\partial x^3} + \frac{\partial^3 w}{\partial x \partial y^2}\right); \quad Q_y = -D\left(\frac{\partial^3 w}{\partial y^3} + \frac{\partial^3 w}{\partial y \partial x^2}\right). \tag{3.26}$$

From the elasticity theory it is possible to calculate the shear stress trough thickness in terms of Q_x and Q_y. Considering a parabolic distribution for the tangential stresses across the plane thickness, the maximum shear stress is given by,

$$(\gamma_{xz})_{max} = \frac{3}{2}\frac{Q_x}{t}; \quad (\gamma_{yz})_{max} = \frac{3}{2}\frac{Q_y}{t}. \tag{3.27}$$

3.6 Boundary conditions

There are some classical boundary conditions that can be imposed on nodes. Depending on the aim of the nodal restriction, different conditions have to be applied:

1. *Fixed boundary*, values of the displacement are specified at the restrained parts of the boundary. This condition is represented as,

   $$w = \overline{w}; \quad \theta_n = \overline{\phi}_n; \quad and \quad \theta_s = \phi_s$$

   n and s are directions normal and tangential to the boundary curve of the middle surface and being $(\bar{\cdot})$ a prescribed value. The special case of *clamped edge* is when zeros values are assigned. Note that for Kirchhoff thin plate elements $\theta_s = \frac{\partial w}{\partial s}$.

2. *Traction boundary* where stress resultants M_n, M_{ns} *and* S_n are given prescribed values,

   $$M_n = \overline{M}_n; \quad M_{ns} = \overline{M}_{ns} \quad and \quad S_n = \overline{S}_n$$

   A *free edge* is a special case where zero values are assigned.

3. *Mixed boundary conditions*, where both traction and displacement values are specified. A common case is the *simply supported edge* (SS), here $w = 0$ therefore $\theta_s = 0$ and $M_n = 0$ and $M_{ns} = 0$.

3.7 Continuity requirements

The presence of second derivatives in the strain representation indicates that C^1 continuity for shape functions is mandatory. To ensure the required continuity of w and its normal slope across an interface both w and $\partial w/\partial n$ must be defined by values of nodal parameters along such interface.

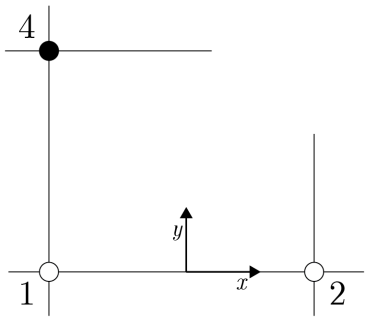


Figure 3.4: Continuity requirements for normal slopes.

Lets consider the line $1-2$ of the represented rectangular plate element, let n be represented in the y direction, it is necessary that w and $\partial w/\partial y$ be uniquely determined by the values of w, $\partial w/\partial x$ and $\partial w/\partial y$ parameters along the same line. The following approximation show C_1 continuity along the depicted side,

$$w = a_0 + a_1 x + a_3 y \ ... \tag{3.28}$$

and

$$\frac{\partial w}{\partial y} = b_0 + b_1 x + b_2 \ ... \tag{3.29}$$

the number of constants in each equation must allow to determine a unique solution for the nodal parameters.

Along the side 1-2 the function $\partial w/\partial y$ depends only on the 1-2 nodal values of the $\partial w/\partial y$ while along the 1-4 side the function $\partial w/\partial x$ depends only on 1-4 nodal values of dw/dx. Having the function $\partial w/\partial y$ to be differentiated with respect to x, along side 1-2 the subsequent function $\partial^2 w/\partial x \partial y$ depends on the nodal values of 1-2 line only, similarly differentiating $\partial w/\partial x$ with respect to y depends uniquely on the nodal parameters of 1-4 line [19]. At the common node, inconsistency arises as it is not possible to have there the necessary identity for continuous function:

$$\frac{\partial^2 w}{\partial x \partial y} \equiv \frac{\partial^2 w}{\partial y \partial x} \tag{3.30}$$

. Meaning that it is not possible to specify simple polynomial expressions for shape functions ensuring full compatibility when only w, $\partial w/\partial x$, $\partial w/\partial y$ are prescribed at corner nodes [24]. Thus, If any function satisfying the compatibility are found with three nodal variables, they must be such that at corner nodes these functions are not continuously differentiable and the cross derivative is not unique. To specify the cross-derivative as one of the nodal degrees of freedom is a way to overcome this difficulty [24]. This, when using rectangular elements mesh, is convenient and permissible. Applying this idea to nodes at which a number of element interfaces are connected with distinct angles, it is generally not permissible.

The difficulties of finding compatible functions led to many attempts of ignoring the complete slope continuity while still continuing with other criteria, lead to the appearing of several non-conforming elements.

3.8 The rectangular plate element

Melosh [17], O. Zienkiewicz and Y. Chung [25,26] developed a 4-noded rectangular plate element known as MCZ plate element represented in the Figure 3.5 and described in the following. Consider a plate element with the middle plane coincident with the xy plane. At each node i, displacements u_i are introduced. This nodal displacement vector is given by three components, the displacement in the z direction w, a rotation about the x axis θ_{xi} and a rotation around y axis θ_{yi}. The element displacement denoted by u_e is given by listing all nodal displacements totalling twelve:

$$u_e = \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{Bmatrix}; \quad \text{with} \quad u_i = \begin{Bmatrix} w_i \\ \theta_{xi} \\ \theta_{yi} \end{Bmatrix}. \tag{3.31}$$

A polynomial approximation is used to define the shape functions in terms of 12 parameters. Note that certain terms must be omitted from a complete fourth-order polynomial.

$$\begin{aligned} w =& \alpha_1 + \alpha_2 x + \alpha_3 y + \alpha_4 x^2 + \alpha_5 xy + \alpha_6 y^2 + \alpha_7 x^3 + \alpha_8 x^2 y \\ &+ \alpha_9 xy^2 + \alpha_{10} y^3 + \alpha_{11} x^3 y + \alpha_{12} xy^3 \\ \equiv& \mathbf{P}\alpha. \end{aligned} \tag{3.32}$$

This representation has certain advantages, along any x constant or y constant line, the displacement will always vary as cubic. Element boundaries are composed of such lines. As such end values are common to adjacent elements continuity of w will be imposed along the interface [24].

The constants α can be obtained by imposing at each node

$$w_i = (w)_i; \quad \frac{\partial w}{\partial x} = \theta_{xi}; \quad \frac{\partial w}{\partial y} = \theta_{yi}, \tag{3.33}$$

where $i = 1, 2, ..., n_{nodes}$. It can be found after some algebra that,

$$\mathbf{a}^{(e)} = [w_1, \theta_{x1}, \theta_{y1}, ..., w_4, \theta_{x4}, \theta_{y4},] = \mathbf{A}[\alpha_1, \alpha_2, \alpha_3, ..., \alpha_{11}, \alpha_{12}]^{\mathrm{T}} \tag{3.34}$$

with,

$$\mathbf{A} = \begin{bmatrix} 1 & x_1 & y_1 & x_1^2 & x_1y_1 & y_1^2 & x_1^3 & x_1^2y_1 & x_1y_1^2 & y_1^3 & x_1^3y_1 & x_1y_1^3 \\ 0 & 1 & 0 & 2x_1 & y_1 & 0 & 3x_1^2 & 2x_1y_1 & y_1^2 & 0 & 3x^2y_1 & y_1^3 \\ 0 & 0 & 1 & 0 & x_1 & 2y_1 & 0 & x_1^2 & 2x_1y_1 & 3y_1^2 & x_1^3 & 2x_1y_1^2 \\ & & & \vdots & & & \vdots & & & \vdots & & \\ & & & \vdots & & & \vdots & & & \vdots & & \\ 0 & 0 & 1 & 0 & x_1 & 2y_1 & 0 & x_1^2 & 2x_1y_1 & 3y_1^2 & x_1^3 & 2x_1y_1^2 \end{bmatrix}. \tag{3.35}$$

From Equation 3.34, it comes

$$\alpha = \mathbf{A}^{-1}\mathbf{a}^{(e)}, \tag{3.36}$$

That, combining with the Equations 3.32 and 3.36 yields,

$$w = \mathbf{P}^{\mathrm{T}}\alpha = \mathbf{P}^{\mathrm{T}}\mathbf{A}^{-1}\mathbf{a}^{(e)} = \mathbf{N}\mathbf{a}^{(e)}, \tag{3.37}$$

being

$$\mathbf{N} = \mathbf{P}^{\mathrm{T}}\mathbf{A}^{-1}, \tag{3.38}$$

the shape function matrix. An explicit form of the shape function was derived by Melosh [17] and can be written in terms of natural coordinates as

$$\mathbf{N} = [N_1 \;\; N_2 \;\; N_3 \;\; N_4] \;\text{ and }\; N_i = [N_i \overline{N}_i \overline{\overline{N}}_i], \tag{3.39}$$

being

$$\begin{aligned} N_i &= (1+\xi_i\xi)(1+\eta_i\eta)(2+\xi_i\xi+\eta_i\eta-\xi^2-\eta^2)/8, \\ \overline{N}_i &= a(\xi^2-1)(\xi+\xi_i)(1+\eta_i\eta)/8, \\ \overline{\overline{N}}_i &= b(\eta^2-1)(\eta-\eta_i)(1+\xi_i\xi)/8, \end{aligned} \tag{3.40}$$

where $(\xi_i,\ \eta_i)$ are the natural coordinates associated with the node i and $(\xi,\ \eta)$ are the natural coordinates, which are related with the physical coordinates. The element deflection can be obtained by,

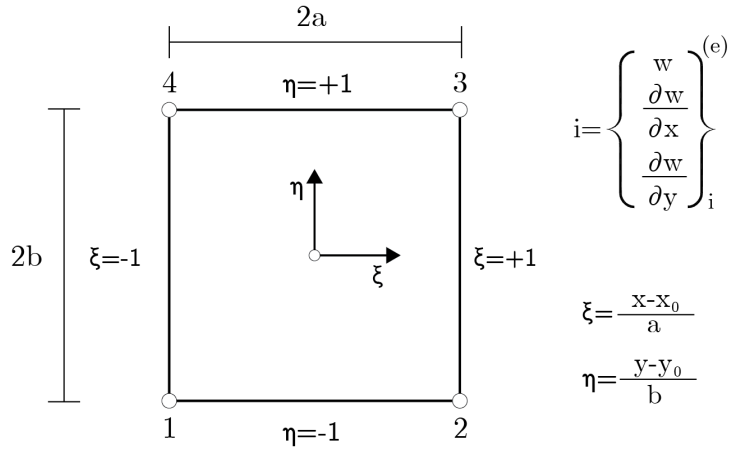


Figure 3.5: Representation of a non-conforming thin plate rectangle.

$$w = \sum_{i=1}^{4}\left[N_i w_i + \overline{N}_i\left(\frac{\partial w}{\partial x}\right) + \overline{\overline{N}}_i\left(\frac{\partial w}{\partial y}\right)\right]. \tag{3.41}$$

3.8.1 Stiffness and load matrices

The strain vector or curvature vector can be represented as,

$$\hat{\varepsilon}_{=} \begin{Bmatrix} \frac{\partial^2 w}{\partial x^2} \\ \frac{\partial^2 w}{\partial y^2} \\ 2\frac{\partial^2 w}{\partial x \partial y} \end{Bmatrix} = \sum_{i=1}^{4} \mathbf{B}_{b_i} \mathbf{a}^{(e)} = \mathbf{B}_b \mathbf{a}^{(e)} \tag{3.42}$$

with

$$\mathbf{B}_b = [B_{b1} B_{b2} B_{b3} B_{b4}] \quad , \quad B_{bi} = \begin{bmatrix} \frac{\partial^2 N_i}{\partial x^2} & \frac{\partial^2 \overline{N}_i}{\partial x^2} & \frac{\partial^2 \overline{\overline{N}}_i}{\partial x^2} \\ \frac{\partial^2 N_i}{\partial y^2} & \frac{\partial^2 \overline{N}_i}{\partial y^2} & \frac{\partial^2 \overline{\overline{N}}_i}{\partial y^2} \\ 2\frac{\partial^2 N_i}{\partial xy} & 2\frac{\partial^2 \overline{N}_i}{\partial xy} & 2\frac{\partial^2 \overline{\overline{N}}_i}{\partial xy} \end{bmatrix} \tag{3.43}$$

The bending moment field is obtained in terms of the nodal degrees of freedom, substituting Equation 3.42 into Equation 3.12,

$$\hat{\sigma}_b = \hat{\mathbf{D}}_b \mathbf{B}_b \mathbf{a}^{(e)}. \tag{3.44}$$

Using the the standard equilibrium formula 2.35 for the element is found that

$$\mathbf{K}^{(e)} \mathbf{a}^{(e)} - \mathbf{f}^{(e)} = \mathbf{q}^{(e)}. \tag{3.45}$$

Where the element stiffness is represented as,

$$\mathbf{K}_{ij}^{(e)} = \iint_{A^{(e)}} \mathbf{B}_{bi} \hat{\mathbf{D}}_b \mathbf{B}_{bj} \, \mathrm{d}x\mathrm{d}y. \tag{3.46}$$

Here, $A^{(e)}$ denotes the element area and $\hat{\mathbf{D}}_b$ the constitutive matrix represented in the equation 3.13. The equivalent nodal force vector for distributed bending moments and loads is

$$\mathbf{f}_i^{(e)} = \begin{Bmatrix} f_{zi} \\ M_{xi} \\ M_{yi} \end{Bmatrix} = \iint_{A^{(e)}} \begin{bmatrix} N_i & N_{i,x} & N_{i,y} \\ \overline{N}_i & \overline{N}_{i,x} & \overline{N}_{i,y} \\ \overline{\overline{N}}_i & \overline{\overline{N}}_{i,x} & \overline{\overline{N}}_{i,x} \end{bmatrix} \, \mathrm{d}x\mathrm{d}y, \tag{3.47}$$

where f_{zi} is the vertical force, M_{xi} and M_{yi} are the bending forces acting in the node i.

Intentionally left blank.

Chapter 4

Spline theory

It is of the utmost importance to comprehend some concepts about geometry representation and also spline theory, being isogeometric analysis a CAD geometry based method. This section presents a basic approach of concepts such as parametric representation, Bézier, B-spline, NURBS (non-uniform rational B-splines) curves, surfaces and its basis functions.

In mathematics curves and surfaces can be represented in three forms, namely explicitly, implicitly or parametrically. Explicit representation of the form $y = f(x)$ for curves and $z = f(x, y)$ for surfaces is the easiest but also the most limited one. It's easy to calculate it's derivative and to obtain geometric information about it, also it is easy to find an intersection with another curve, however it is very axis dependent, for each value of x there is only one correspondent value of y so for this reason it is the least used form of representation in computer aided design.

Implicit representation, on the other hand, takes the form of $f(x, y) = 0$ for curves and $f(x, y, z) = 0$ for surfaces, it is a more versatile representation as it works in order to find a solution to a set equation it can have one more value on each axis, and like explicit representation it is also easy to know if a point is in the curve and also to know its derivative and, although being rather difficult to find a intersection with another curve and still axis dependent, it has a variety of uses in computer aided design [27].

Parametric curve and surface geometric representations are fundamental in order to understand the concepts of Bézier and B-spline curve and surface definition as both are parametrically represented [2, 3].

4.1 Parametric representation

Parametric representation takes the form of

$$x = f(t);\ y = g(t);\ z = h(t)\ , \tag{4.1}$$

where t is a parameter defined in a finite interval and the equations f, g and h map the parameter t in the parametric space to a point x, y, z in the physical space. It is axis independent, as it can have various points of the curve in the same axis, and also has additional degrees of freedom compared to other formulations [2].

For comparison purposes, first take a curve defined with explicit formulation where four degrees of freedom are present one for each coefficient a,b,c and d.

$$y = ax^3 + bx^2 + cx + d\ , \tag{4.2}$$

rewriting the same equation in a parametric form,

$$x(t) = \alpha t^3 + \beta t^2 + \gamma t + \delta, \tag{4.3}$$

$$y(t) = \overline{\alpha} t^3 + \overline{\beta} t^2 + \overline{\gamma} t + \overline{\delta}\ . \tag{4.4}$$

Like in explicit form, there is one degree of freedom for each coefficient whereas in this case there are eight: $\alpha,\ \beta,\ \gamma,\ \delta,\ \overline{\alpha},\ \overline{\beta},\ \overline{\gamma},\ \overline{\delta}$,

To calculate the derivative of y with respect to x,

$$\frac{\partial y}{\partial x} = \frac{\frac{\partial y}{\partial t}}{\frac{\partial x}{\partial t}}, \tag{4.5}$$

using, as a example the equation above, its derivative is equal to

$$\frac{\partial y}{\partial x} = \frac{3\overline{\alpha} t^2 + 2\overline{\beta} t + \overline{\gamma}}{3\alpha t^2 + 2\beta t + \gamma}. \tag{4.6}$$

When the denominator is zero, the derivative is infinite which means that, geometrically, it corresponds to a vertical edge.

A parametric curve extended to three dimension space is obtained by specifying $z(t) = h(t)$, where again t is the parameter. Any point that is on a geometry of a parametric representation is obtained by the vector-valued function

$$P(t) = [x(t)\ y(t)\ z(t)] = [f(t)\ g(t)\ h(t)]\ , \tag{4.7}$$

function that has its components *x, y, z* that share the same parameter t but mapped independently, which means that *f, g, h* can be different equations [3].

4.1.1 A parametric line

Considering P_1 and P_2 two vector-valued functions or position vectors, and t a parameter, a simple general parametric equation for a straight line segment is

$$P(t) = P_1 + (P_2 - P_1)t \quad 0 \leq t \leq 1\ . \tag{4.8}$$

A position vector has components in some coordinate system. In a Cartesian system (x, y, z) the representation is $P = [x \quad y \quad z]$. Each component of the position vector has also a parametric representation $x(t),\ y(t),\ z(t)$.

4.1.2 A parametric surface

A parametric surface requires two parameters to be represented,

$$x = x(u, v) \quad y = y(u, v) \quad z = z(u, v). \tag{4.9}$$

A surface is said to be biparametric. Notice that if one parameter is held constant, the result is a curve on the surface. If both parameters are held constant then a point on the surface is formed. The maximum and minimum of each parameter represents the edges of the surface.

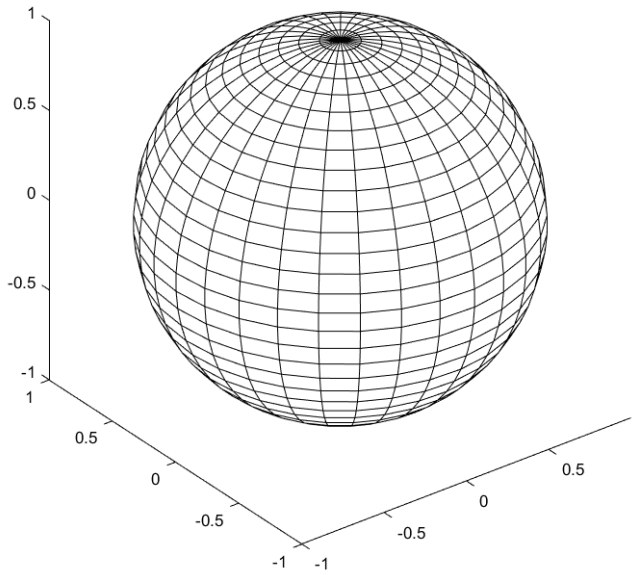


Figure 4.1: Representation of a parametric sphere.

Figure 4.1 is an example of a parametric surface, being the geometric representation of a sphere where $x = rcos(\theta)cos(\phi)$, $y = rcos(\theta)rsin(\phi)$ and $z = rcos(\phi)$ with $r = 1$, $0 \leq \theta \leq 2\pi$ and $0 \leq \phi \leq \pi$.

The partial derivatives of a known surface

$$Q(u, w) = Q(x(u, w), y(u, w), z(u, w)), \tag{4.10}$$

are defined by

$$Q_u(u, w) = Q(x_u(u, w), y_u(u, w), z_u(u, w)) \tag{4.11}$$

and

$$Q_w(u, w) = Q(x_w(u, w), y_w(u, w), z_w(u, w) \quad , \tag{4.12}$$

denote that the cross product between both at a point gives the unit normal vector n.

$$n = \frac{Q_u \times Q_w}{|Q_u \times Q_w|} \quad |Q_u \times Q_w| \neq 0, \tag{4.13}$$

where $Q_u = \partial Q/\partial u$ and $Q_w = \partial Q/\partial w$.

4.2 Bézier curves

In this section it will be done an introduction to Bézier curves. B-splines emerged from the Bézier curves, and NURBS from B-splines, and so many properties are inherent.

The most common applications involve CAD software and 3D modeling. A Bézier curve is determined by what is called a control polygon, an n degree Bézier curve is defined by $n + 1$ control points .

4.2.1 Bézier curves definition

Mathematically, the general form of a n degree Bézier curve is defined by

$$P(t) = \sum_{i=0}^{n} B_i J_{n,i}(t), \quad 0 \leq t \leq 1 \tag{4.14}$$

where B_i is the matrix with the coordinates for control points and $J_{n,i}(t)$ is called the *nth*-degree Bernstein or Bézier basis function.

The basis functions $(J_{i,n})$ are given by

$$J_{i,n}(t) = \begin{pmatrix} n \\ i \end{pmatrix} t^i (1-t)^{n-i}, \quad (0)^0 \equiv 1 \tag{4.15}$$

being $\begin{pmatrix} n \\ i \end{pmatrix}$ the binomial coefficient established by

$$\begin{pmatrix} n \\ i \end{pmatrix} = \frac{n!}{i!(n-i)!}, \quad 0! \equiv 1, \tag{4.16}$$

a equation that is read as "n choose i", since it gives the number of ways of choosing i items from a set of n, and $n!$ the factorial of n. Here n is the degree of the Bernstein basis hence the number of curve segments, and being the vertices numbered from 0 to n, n also is the value of the last control vertex as represented in the Figure 4.2 [2, 3].

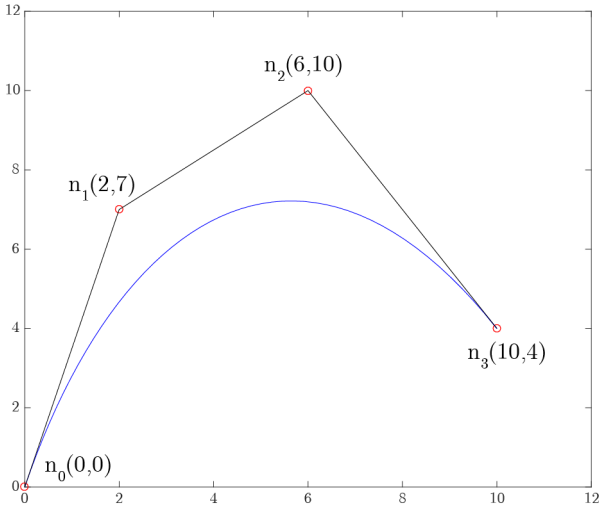


Figure 4.2: Cubic Bézier Curve and its control polygon.

4.2.2 Bézier curve properties

There are some fundamental properties concerning the comprehension of Bézier curves that are to keep in mind. Bézier curves are defined by Bernstein basis and that gives them inherent properties [3]:

1. Basis function are real;

2. The defining polynomial degree is one less than the number of control points;
3. The start and end points of the curve are coincident with the first and last points of the control polygon;
4. The curve is always within the convex hull defined by the control polygon;
5. The curve is invariant under an affine transformation.

4.2.3 Bézier curve derivatives

Recalling the definition of Bézier curve, represented in the equation 4.14, the first derivative is determined by

$$P'(t) = \sum_{i=0}^{n} B_i J'_{n,i}(t), \quad 0 \leq t \leq 1, \tag{4.17}$$

while the second derivative is given by

$$P''(t) = \sum_{i=0}^{n} B_i J''_{n,i}(t), \quad 0 \leq t \leq 1 \quad . \tag{4.18}$$

The basis function derivatives are obtained by

$$\begin{aligned} J'_{n,i}(t) &= \begin{pmatrix} n \\ i \end{pmatrix} (it^{i-1}(1-t)^{n-i} - (n-i)t^i(1-t)^{n-i-1}) \\ &= \begin{pmatrix} n \\ i \end{pmatrix} t^i(1-t)^{n-i} \left(\frac{i}{t} - \frac{n-i}{1-t}\right) \\ &= \frac{i-nt}{t(1-t)} J_{n,i}(t). \end{aligned} \tag{4.19}$$

Analogously, the second derivative comes as

$$J''_{n,i}(t) = \left(\frac{(i-nt)^2 - nt^2 - i(1-2t)}{t^2(1-t)^2}\right) J_{n,i}(t). \tag{4.20}$$

With both derivatives it is possible to determine the curve normals or local curvature [3].

4.2.4 Bézier surfaces definition

A tensor product Bézier surface has the form of

$$Q(u,w) = \sum_{i=0}^{n} \sum_{j=0}^{m} B_{i,j} J_{n,i}(u) K_{m,j}(w), \tag{4.21}$$

where $J_{n,i}(u)$ and $K_{m,j}(w)$ are basis function in the u and w directions [3], note that the (u,w) domain of this mapping is a rectangle [2]. $B_{i,j}$ are the vertices of the control

polygon. Repeating the definition of Bernstein basis given by Equation 4.15, in order to define $J_{n,i}(u)$ and $K_{m,j}(w)$

$$J_{n,i}(u) = \begin{pmatrix} n \\ i \end{pmatrix} u^i (1-u)^{n-i}, \quad (0)^0 \equiv 1, \tag{4.22}$$

where,

$$\begin{pmatrix} n \\ i \end{pmatrix} = \frac{n!}{i!(n-i)!}, \quad 0! \equiv 1, \tag{4.23}$$

and similarly

$$K_{m,j}(w) = \begin{pmatrix} m \\ j \end{pmatrix} w^j (1-w)^{m-j}, \quad (0)^0 \equiv 1, \tag{4.24}$$

with,

$$\begin{pmatrix} m \\ j \end{pmatrix} = \frac{m!}{j!(m-j)!}, \quad 0! \equiv 1\ . \tag{4.25}$$

As well as the Bézier curve representation, the index n and m are equal to the number of control vertices minus one in u and w directions respectively. The degree of the surface in each direction is one less than the number of control vertices [3] as example the Figure 4.3 represents a cubic Bézier surface made with 4 by 4 control vertices.

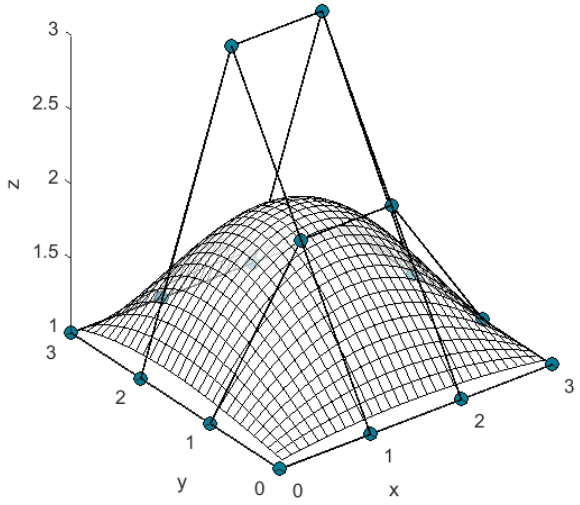


Figure 4.3: Bézier surface and its control polygon.

4.2.5 Bézier surface properties

Again, as well as Bézier curves, Bézier surfaces are defined by Bernstein basis, and that gives them known properties [2], which are important to enunciate. A Bézier surface is restricted by the following characteristics:

1. Non negativity: $J_{n,i}(u)K_{m,j}(w) \geq 0$ for all i, j, u, v;

2. $\sum_{i=0}^{n}\sum_{j=0}^{m} J_{n,i}(u)K_{m,j}(w) = 1,\ \forall u, v$;
3. $Q_{u,w}$ is within the convex hull created by its control polygon;
4. The surface interpolates the four corner control points;
5. Transformation invariance;
6. Variation diminishing property for Bézier surface is not known.

4.2.6 Bézier surface derivatives

The derivatives of Bézier surface are also obtained through formal differentiation. Using the Equation 4.21 that defines a Bézier surface, the first and second parametric partial derivative are determined, respectively, as [3],

$$Q_u(u,w) = \sum_{i=0}^{n}\sum_{j=0}^{m} B_{i,j}J'_{n,i}(u)K_{m,j}(w), \tag{4.26}$$

$$Q_w(u,w) = \sum_{i=0}^{n}\sum_{j=0}^{m} B_{i,j}J_{n,i}(u)K'_{m,j}(w), \tag{4.27}$$

$$Q_{u,w}(u,w) = \sum_{i=0}^{n}\sum_{j=0}^{m} B_{i,j}J'_{n,i}(u)K'_{m,j}(w), \tag{4.28}$$

$$Q_{uu}(u,w) = \sum_{i=0}^{n}\sum_{j=0}^{m} B_{i,j}J''_{n,i}(u)K_{m,j}(w), \tag{4.29}$$

$$Q_{w,w}(u,w) = \sum_{i=0}^{n}\sum_{j=0}^{m} B_{i,j}J_{n,i}(u)K''_{m,j}(w), \tag{4.30}$$

where Q_u represents $\partial Q/\partial u$. Likewise, $Q_u w$ represents $\partial^2 Q/\partial uw$. The derivatives $J'_{n,i}$, $J''_{n,i}$,$K'_{m,j}$ and $K''_{m,j}$ are obtained by Equations 4.19 and 4.20.

4.3 B-splines

Bézier curves and surfaces have two shortcomings: first when the number of control points increases the degree of a curve increases. To decrease the degree of the curve, the number of vertices has to decrease, as well, and this creates a dependency. Second, when a value on a control point is changed, the result is felt in all curve, for the reason that any point on the curve is an outcome of the blending of values form all control points, this eliminates the ability to locally control the curve.

B-spline basis (from basis spline), has non-global behavior which means that each vertex B_i has only one correspondent basis function. Each vertex influences the curve geometry only for a range of parameters where the associated basis is non-zero. The degree of a B-spline can be changed without changing the number of control vertices. The theory for B-splines was first introduced in [28].

4.3.1 B-Spline curves definition

A B-spline curve is geometrically defined by its basis function $N_{i,k}$ and the $n+1$ control points position vector B_i. Given $P(t)$, a position vector along the curve as function of the parameter t, the resulting B-spline is represented by

$$P(t) = \sum_{i=1}^{n+1} B_i N_{i,k}(t) \quad t_{min} \leq t \leq t_{max}, \quad 2 \leq k \leq n+1. \tag{4.31}$$

A B-spline curve function is considered to be a polynomial spline function of order k because it formally meet the following condition [3]:

P(t) is a polynomial of degree $k-1$ on each interval $x_i \leq t \leq x_{i+1}$.

P(t) and its derivatives of order 1,2,..,$k-2$ are all continuous over the entire curve.

A fourth order B-spline curve is a three order piecewise cubic curve.

4.3.2 Basis functions

The way to define B-spline basis is by a recursion formula developed by Cox [29] and by de Boor [30] and later on, applied as B-spline basis by Riesenfeld [31] and Gordon and Riesinfeld [32], called Cox-de Boor recursion formula. The *ith* normalized B-spline basis function of k-order (degree $p = k-1$), designated $N_{i,k}(t)$, defined as,

$$N_{i,1}(t) = \begin{cases} 1 & \text{if } x_i \leq t \leq x_{i+1} \\ 0 & \text{otherwise} \end{cases}, \tag{4.32}$$

and,

$$N_{i,k}(t) = \frac{t - x_i}{x_{i+k-1} - x_i} N_{i,k-1}(t) + \frac{x_{i+k} - t}{x_{i+k} - x_{i+1}} N_{i+1,k-1}(t), \tag{4.33}$$

where $X = [x_0, ..., x_m]$ is a non-decreasing sequence of real numbers, x_i are called knots and X knot vector.

The knot vector is shown to be of great importance in the definition of the b-spline basis and therefore on the corresponding curve. As already said, it is a non-decreasing array. There are two types of knot vectors, periodic and open, which divide into uniform and non-uniform. A uniform knot vector is given by a sequence of evenly spaced real numbers $X = [0\ 1\ 2\ 3\ 4\ 5]$ and if normalized, parametric values are in the range between 0 and 1 ($x \in [0, ..., 1]$), for example, $X = [0\ 0.25\ 0.5\ 0.75\ 1]$. An open knot vector has end knot values with multiplicity equal to the order of the curve and are given by:

$$\begin{array}{ll} x_i = 0 & 1 \leq i \leq k \\ x_i = i - k & k+1 \leq i \leq n+1 \\ x_i = n - k + 2 & n+2 \leq i \leq n+k+1 \end{array},$$

therefore, given a 3^{th} order B-spline, an example of an open uniform knot vector has

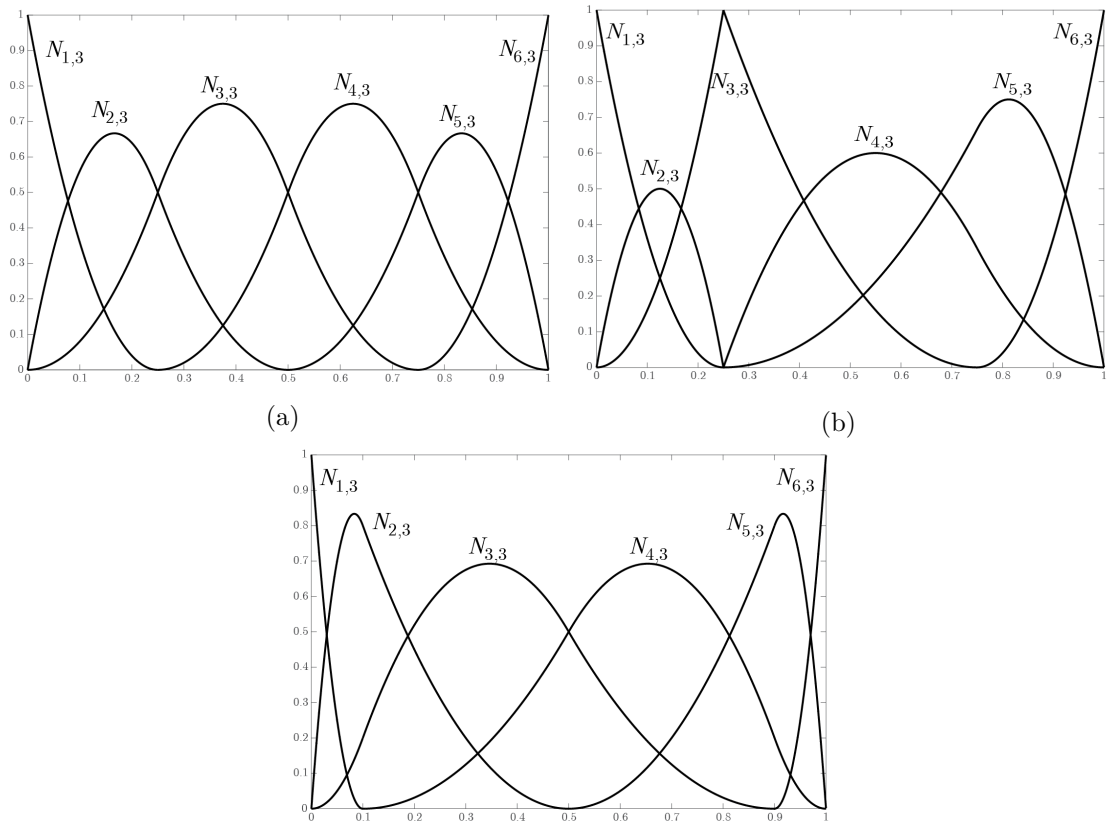


Figure 4.4: Example of B-spline basis function for $n+1=6$ control points with degree of $k=3$: (a) X=[0 0 0 0.25 0.50 0.75 1 1 1] (b) X=[0 0 0 0.25 0.25 0.75 1 1 1] (c) X=[0 0 0 0.10 0.50 0.90 1 1 1]

the form of X = [0 0 0 1 2 3 4 4 4] or if normalized X = [0 0 0 0.25 0.5 0.75 1 1 1] as represented in the Figure 4.4 (a).

There is also other concept that is important to comprehend, which is the internal knot multiplicity and also the usage of multiple control points. These are ways that can be used to control a B-spline.

Considering the Figure 4.5 that represents $k=4$ order B-spline curves. One curve with four control points and no multiple control vertices, here the knot vector is X = [0 0 0 0 1 1 1 1]. The intermediate curve is defined by five control points with two multiple knot vertices on the point [2 7] here the knot vector is given by [0 0 0 0 1 2 2 2 2]. The other curve is defined by six polygon vertices at three multiple control points at [2 7], its knot vector is [0 0 0 0 1 2 3 3 3 3].

Notice that as the number of multiple vertices increases at [2 7] the curve is pulled closer to the same point. When the number of multiple vertices reaches $k-1$ a sharp corner or cusp is created. Although a cusp is formed in the highest curve, the C^{k-2} differentiability of the curve is maintained. The perk to include sharp corner differentiable is a important characteristic of B-spline curves.

Figure 4.6 represents the effect of multiple interior knot values of a third order curves

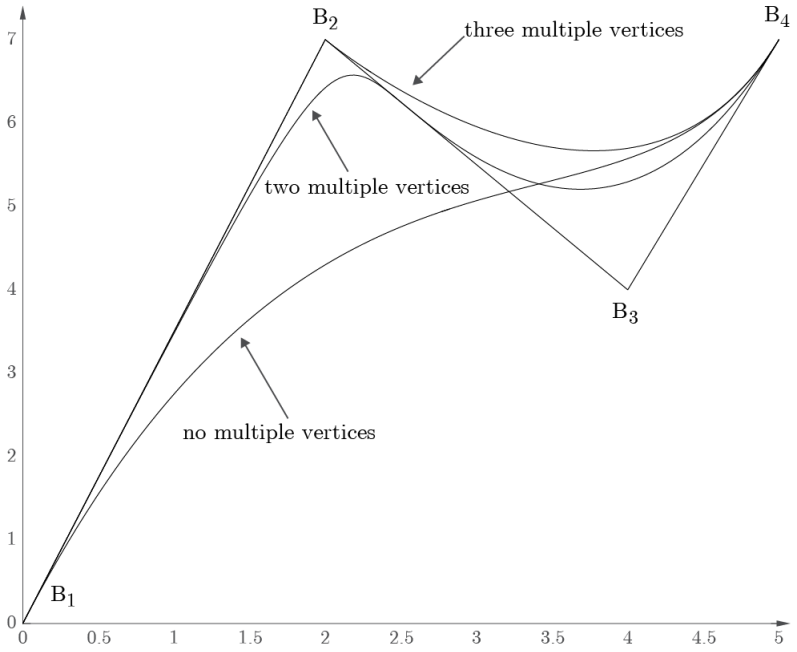


Figure 4.5: Effect of multiple polygon vertices.

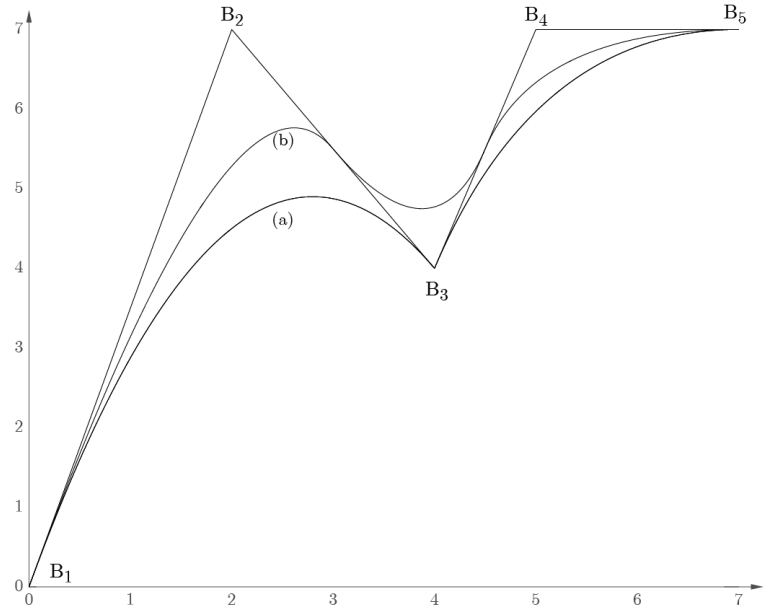


Figure 4.6: Effect of multiple interior knot values.

$k = 3$. Curve (a) has a non-uniform knot vector with multiple interior knots and is given by [0 0 0 1 1 3 3 3]. An identical curve is obtained with the knot vector defined as [0 0 0 2 2 3 3 3]. While curve (b) has an uniform knot vector which is [0 0 0 1 2 3 3 3].

Note that curve (a) has a cusp at $B_3 = [4\ 4]$ yielded by the multiple interior knot. A multiple knot introduces a span of zero length which reduces the width of support of the basis function. In contrast to the control points multiplicity, the differentiability of the basis at x_i is reduced to C^{k-m-1} where $m \leq k-1$ is the multiplicity of the interior knot value. In the figure the non-uniform curve is C^0 continuous near the sharp corner hence, a discontinuity occurs.

4.3.3 B-spline curve properties

For easer comprehension, it is important to clarify fundamental B-spline proprieties and its behaviour in special cases. Again, B-spline curves are defined by basis functions $N_{i,k}$, let x_i be the *ith* knot, a B-spline curve is defined by the following:

1. The maximum order of a B-spline equals the number of its control vertices. When at its maximum order, a B-spline is identical to a Bézier curve;

2. $N_{i,1}(t)$ Is only different than zero on the half-open interval $x \in [x_i\ ,\ x_{i+1}[$;

3. For $k > 1$, $N_{i,k}$ is a linear combination of two basis function with *k-1* order;

4. A *ith knot span* is defined by the interval $[x_i,\ x_{i+1}[$;

5. B-splines basis functions are a partition of unity;

$$\sum_{i=0}^{n} B_{i,p}(t) = 1 \quad \forall t \in [a, b]$$

6. a knot span can have zero length since knots doesn't have to be distinct;

7. the computation of a *k-th* order basis function generates a triangular pattern as seen bellow.

$$\begin{array}{llll} N_{i,k} & & & \\ N_{i,k-1} & N_{i+1,k-1} & & \\ N_{i,k-2} & N_{i+1,k-2} & N_{i+2,k-2} & \\ \vdots & & & \vdots \\ N_{i,1} & N_{i+1,1} & N_{i+2,k} & N_{i+3,1} \end{array}$$

4.3.4 B-Spline curve derivatives

Let the P'(t) be the first derivative of a B-spline curve P(t), the derivative is obtained by formal differentiation:

$$\mathrm{P}'(t) = \sum_{i=1}^{n+1} B_i N'_{i,k}(t) \tag{4.34}$$

where the derivative of the basis function $N'_{i,k}(t)$ is calculated using,

$$\begin{aligned} N'_{i,k}(t) = \frac{1}{x_{i+k-1} - x_i} N_{i,k-1}(t) + \frac{t - x_i}{x_{i+k-1} - x_i} N'_{i,k-1}(t) + \\ + \frac{x_{i+k}}{x_{i+k} - x_{i+1}} N'_{i+1,k-1}(t) - \frac{1}{x_{i+k} - x_{i+1}} N_{i+1,k-1}(t), \end{aligned} \tag{4.35}$$

as seen, it is also a formal differentiation, and considering the equation above it is easy to figure that $N'_{i,1}(t) = 0$, and with $k = 2$ the corresponding basis function derivative is,

$$N'_{i,2}(t) = \frac{1}{x_{i+1} - x_i} N_{i,1}(t) - \frac{1}{x_{i+k} - x_{i+1}} N_{i+1,k-1}(t). \tag{4.36}$$

4.3.5 B-spline surfaces definition

B-spline surfaces are useful for representing surfaces that require smoothness and fairness, e.g., aircraft wings, boat hulls, automobile bodies. The B-spline surface is a special case of Non-Uniform Rational B-splines (NURBS) thus it is not as flexible [3].

A B-spline is obtained trough two knot vectors, which are represented in a bidirectional net of control points. B-spline surface is the extension of a Bézier surface and it is represented by

$$Q(u, w) = \sum_{i=1}^{n+1} \sum_{j=1}^{m+1} B_{i,j} N_{i,k}(u) M_{j,l}(w), \tag{4.37}$$

with $N_{i,k}(u)$ and $M_{j,l}(w)$ being the B-spline basis function in the bi-parametric space u and v directions, respectively. Recalling the definition of basis function given in equations 4.32 and 4.33, for surface definition,

$$\begin{aligned} N_{i,1}(u) &= \begin{cases} 1 & \text{if } x_i \leq u \leq x_{i+1} \\ 0 & \text{otherwise} \end{cases} \\ N_{i,k}(u) &= \frac{u - x_i}{x_{i+k-1} - x_i} N_{i,k-1}(u) + \frac{x_{i+k} - u}{x_{i+k} - x_{i+1}} N_{i+1,k-1}(u), \end{aligned} \tag{4.38}$$

and,

$$\begin{aligned} M_{j,1}(w) &= \begin{cases} 1 & \text{if } y_j \leq w \leq y_{j+1} \\ 0 & \text{otherwise} \end{cases} \\ M_{j,l}(w) &= \frac{w - y_j}{y_{j+l-1} - y_i} M_{j,l-1}(w) + \frac{y_{j+l} - w}{y_{j+l} - y_{j+1}} M_{j+1,l-1}(w), \end{aligned} \tag{4.39}$$

where x_i and y_i are knot vectors, again $B_{i,j}$ the control vertices coordinates. The indices $n \times m$ are one less than the number of control vertices in each direction.

4.3.6 B-spline surface properties

The B-spline surface properties are related to B-spline basis as well as B-spline curves therefore, there are already several known properties. Keeping the formalities used in the the last section its properties are:

1. The maximum order in each direction is equal to the number of control vertices in that direction;

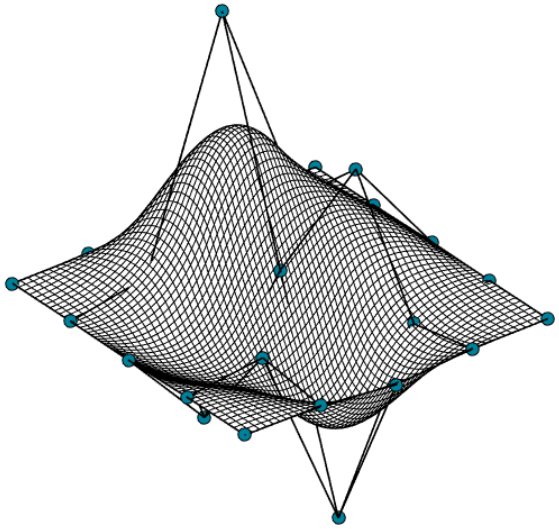


Figure 4.7: B-spline surface and its control polygon.

2. The continuity of the surface in each parametric direction is two less than the order in the same direction;

3. The variation diminishing property is also unknown;

4. The influence of a single control net is restricted to $\pm k/2, \pm l/2$;

5. The surface is within the convex hull of the control polygon net.

4.3.7 B-spline surface derivatives

The parametric derivatives of B-spline surface are obtained in a similar way to Bázier surfaces. Using the equation 4.37 and through formal differentiation, its derivatives first and second derivatives are determined as,

$$Q_u(u,w) = \sum_{i=1}^{n+1}\sum_{j=1}^{m+1} B_{i,j} N'_{i,k}(u) M_{j,l}(w), \tag{4.40}$$

$$Q_w(u,w) = \sum_{i=1}^{n+1}\sum_{j=1}^{m+1} B_{i,j} N_{i,k}(u) M'_{j,l}(w), \tag{4.41}$$

$$Q_{u,w}(u,w) = \sum_{i=1}^{n+1}\sum_{j=1}^{m+1} B_{i,j} N'_{i,k}(u) M'_{j,l}(w), \tag{4.42}$$

$$Q_{uu}(u,w) = \sum_{i=1}^{n+1}\sum_{j=1}^{m+1} B_{i,j} N''_{i,k}(u) M_{j,l}(w), \tag{4.43}$$

$$Q_{w,w}(u,w) = \sum_{i=1}^{n+1}\sum_{j=1}^{m+1} B_{i,j}N_{i,k}(u)M''_{j,l}(w). \tag{4.44}$$

Again, $Q_u = \partial Q/\partial u$ and $Q_{uw} = \partial^2 Q/\partial uw$ the same is valid to the other parametric derivatives. The derivatives of B-spline basis function is given by the Equation 4.35.

4.4 Non-Uniform Rational B-Spline Curves (NURBS)

Rational B-spline provide a precise mathematical form of representing common geometries such as conic curvature, circles, planes and lines, and free-form curves. This type of formulation is the most used in computer aided design.

4.4.1 NURBS curves definition

A rational B-spline curve is defined in four-dimensional (4D) homogeneous coordinate space into a three-dimensional physical space. A *kth* degree curve is defined by,

$$P(t) = \sum_{i=1}^{h} B_i^h N_{i,k}(t), \tag{4.45}$$

where B_i^h are the four-dimensional homogeneous control polygon vertices. $N_{i,k}(t)$ is the non-rational B-spline basis function. The transformation back to the three-dimensional space, is given by dividing through the homogeneous coordinate,

$$P(t) = \frac{\sum\limits_{i=1}^{n+1} B_i h_i N_{i,k}(t)}{\sum\limits_{i=1}^{n+1} h_i N_{i,k}(t)} = \sum_{i=1}^{n+1} B_i R_{i,k}(t), \tag{4.46}$$

where B_i are the control polygon vertices, $R_{i,k}(t)$ are rational B-spline basis functions and h_i are the homogeneous weighting factors.

$$R_{i,k}(t) = \frac{h_i N_{i,k}(t)}{\sum\limits_{i=1}^{n+1} h_i N_{i,k}(t)}, \tag{4.47}$$

here, $h_i \geq 0$ for all values of i.

4.4.2 NURBS curve properties

Rational B-spline functions and curves are a generalization of non-rational splines basis and functions. The properties and geometric characteristics are similar to B-spline.

1. Rational basis function are non-negative $R_{i,k} \geq 0$;
2. The sum of the rational B-spline basis function fora any parameter is always unitary;

$$\sum_{i=1}^{n+1} \equiv 1 \tag{4.48}$$

3. Each basis function has one maximum value with exception to the first degree ($k = 1$);

4. The maximum order of a rational B-spline is one less than the total number of control polygon vertices;

5. For $h_i > 0$, a rational B-spline curve lies within the convex hulls formed by k successive control vertices.

It is clear that when the weights $h_i = 1$ the result curve is a B-spline, whereas it is possible to refer to B-spline as a special case of Rational B-splines.

4.4.3 NURBS curve derivatives

Derivatives of rational functions are difficult to figure, it involves high power denominators [3]. Derivatives of rational B-spline curves are obtained by differentiation and, as B-splines, its basis are as well. The first derivative of a NURBS curve can be determined as,

$$P'(t) = \sum_{i=1}^{n+1} B_i R'_{i,k}(t), \tag{4.49}$$

where,

$$R'_{i,k}(t) = \frac{h_i N'_{i,k}(t)}{\sum_{i=1}^{n+1} h_i N_{i,k}} - \frac{h_i N_{i,k} \sum_{i=1}^{n+1} h_i N'_{i,k}}{\left(\sum_{i=1}^{n+1} h_i N_{i,k}\right)^2}, \tag{4.50}$$

considering $t = 0$ and $t = n - k + 2$, the output is,

$$P'(0) = (k-1)\frac{h_2}{h_1}(B_2 - B_1), \tag{4.51}$$

$$P'(n-k+2) = (k-1)\frac{h_n}{h_{n+1}}(B_{n+1} - B_n), \tag{4.52}$$

which shows that the slope direction in the beginning and the end of the curve is along with the first and last polynomial spans, respectively.

4.4.4 NURBS surfaces definition

Rational B-spline surfaces are the predominant method for surface modelling, as from special cases, non-rational B-splines and Bézier surfaces can be drawn. It gives an excellent local and global control on the knots, and also it can be used to design cylinders, spheres and ellipsoids of revolution. The Cartesian product of a B-spline surface is given by [3],

$$Q(u,w) = \sum_{i=1}^{n+1} \sum_{j=1}^{m+1} B^h_{i,j} N_{i,k}(u) M_{j,l}(w) \tag{4.53}$$

$B^h_{i,j}$ are four dimensional homogeneous control vertices, and $N_{i,k}$ and $M_{j,l}$ are basis functions. Converting into three-dimensional space by dividing through the homogeneous coordinate, comes that

$$Q(u,w) = \frac{\sum_{i=1}^{n+1}\sum_{j=1}^{m+1} h_{i,j}B_{i,j}N_{i,k}(u)M_{j,l}(w)}{\sum_{i=1}^{n+1}\sum_{j=1}^{m+1} h_{i1,j1}N_{i1,k}(u)M_{j1,l}(w)} = \sum_{i=1}^{n+1}\sum_{j=1}^{m+1} B_{i,j}R_{i,j}(u,w), \tag{4.54}$$

where, $B_{i,j}$ are the three-dimension control vertices, and $S_{i,j}(u,w)$ are the bivariate NURBS surface basis function,

$$R_{i,j}(u,w) = \frac{h_{i,j}N_{i,k}(u)M_{j,l}(w)}{\sum_{i=1}^{n+1}\sum_{j=1}^{m+1} h_{i1,j1}N_{i1,k}(u)M_{j1,l}(w)} = \frac{h_{i,j}N_{i,k}(u)M_{j,l}(w)}{Sum(u,w)}, \tag{4.55}$$

being,

$$Sum(u,w) = \sum_{i=1}^{n+1}\sum_{j=1}^{m+1} h_{i1,j1}N_{i1,k}(u)M_{j1,l}(w), \tag{4.56}$$

it is to assume that $h_{i,j} \geq 0 \ \forall \ i,j$.

4.4.5 NURBS surface properties

Rational B-spline properties are similar in terms of analytic and geometric properties to their non-rational counterparts. Some of fundamentals are,

1. The sum of rational basis function for any u and w is

$$\sum_{i=1}^{n+1}\sum_{j=1}^{m+1} R_{i,j}(u,w) \equiv 1 \ ; \tag{4.57}$$

2. The maximum order of a rational B-spline in each direction is equal to the number of control points in the same direction;
3. A rational B-spline surface of order k, l is C^{k-2}, C^{l-2} everywhere;
4. The variation diminishing property is also unknown for rational B-splines;
5. The influence of a control points is $\pm k/2$ $\pm l/2$ spans in each parametric direction;
6. The surface lies in the convex hull of the control net formed by taking the union of all convex hulls of k, l neighbouring the control net vertices, for $h_{i,j} \geq 0$;
7. If the number of net vertices is equal to the order in each parametric direction and there are no duplicate knots, the resulting rational B-spline surface is a rational Bézier surface.

from Equations 4.54 and 4.55 if $h_{i,j} = 1$ the resulting rational B-spline is reduced to its non-rational counterpart.

4.4.6 NURBS surface and basis derivatives

The derivatives of rational B-spline surfaces are obtained through formal differentiation of equation 4.54,

$$Q_u = \frac{\overline{N}}{\overline{D}}\left(\frac{\overline{N}_u}{\overline{N}} - \frac{\overline{D}_u}{\overline{D}}\right), \tag{4.58}$$

$$Q_w = \frac{\overline{N}}{\overline{D}}\left(\frac{\overline{N}_w}{\overline{N}} - \frac{\overline{D}_w}{\overline{D}}\right), \tag{4.59}$$

$$Q_{u,w} = \frac{\overline{N}}{\overline{D}}\left(\frac{\overline{N}_{uw}}{\overline{N}} - \frac{\overline{N}_u}{\overline{N}}\frac{\overline{D}_w}{\overline{D}} - \frac{\overline{N}_w}{\overline{N}}\frac{\overline{D}_u}{\overline{D}} + 2\frac{\overline{D}_u}{\overline{D}}\frac{\overline{D}_w}{\overline{D}} - \frac{\overline{D}_{uw}}{\overline{D}}\right), \tag{4.60}$$

$$Q_{u,u} = \frac{\overline{N}}{\overline{D}}\left(\frac{\overline{N}_{uu}}{\overline{N}} - 2\frac{\overline{N}_u}{\overline{N}}\frac{\overline{D}_u}{\overline{D}} + 2\frac{\overline{D}_u^2}{\overline{D}^2} - \frac{\overline{D}_{uu}}{\overline{D}}\right), \tag{4.61}$$

$$Q_{w,w} = \frac{\overline{N}}{\overline{D}}\left(\frac{\overline{N}_{ww}}{\overline{N}} - 2\frac{\overline{N}_w}{\overline{N}}\frac{\overline{D}_w}{\overline{D}} + 2\frac{\overline{D}_w^2}{\overline{D}^2} - \frac{\overline{D}_{ww}}{\overline{D}}\right). \tag{4.62}$$

Similarly to the other curve derivatives, $Q_u = \partial Q/\partial u$ and for the second derivative $Q_{uw} = \partial^2 Q/\partial uw$. The represented $\overline{N}$ and $\overline{D}$ are numerator and denominator of Equation 4.54, respectively. Numerator derivatives can be obtained as follows,

$$\overline{N}_u = \sum_{i=1}^{n+1}\sum_{j=1}^{m+1} h_{i,j} B_{i,j} N'_{i,k}(u) M_{j,l}(w), \tag{4.63}$$

$$\overline{N}_w = \sum_{i=1}^{n+1}\sum_{j=1}^{m+1} h_{i,j} B_{i,j} N_{i,k}(u) M'_{j,l}(w), \tag{4.64}$$

$$\overline{N}_{uw} = \sum_{i=1}^{n+1}\sum_{j=1}^{m+1} h_{i,j} B_{i,j} N'_{i,k}(u) M'_{j,l}(w), \tag{4.65}$$

$$\overline{N}_{uu} = \sum_{i=1}^{n+1}\sum_{j=1}^{m+1} h_{i,j} B_{i,j} N''_{i,k}(u) M_{j,l}(w), \tag{4.66}$$

$$\overline{N}_{ww} = \sum_{i=1}^{n+1}\sum_{j=1}^{m+1} h_{i,j} B_{i,j} N_{i,k}(u) M''_{j,l}(w), \tag{4.67}$$

Note that the denominator is equivalent to the $Sum(u,w)$ presented in Equation 4.56, its derivatives can be obtained by

$$\overline{D}_u = \sum_{i=1}^{n+1}\sum_{j=1}^{m+1} h_{i,j} N'_{i,k}(u) M_{j,l}(w), \tag{4.68}$$

$$\overline{D}_w = \sum_{i=1}^{n+1}\sum_{j=1}^{m+1} h_{i,j} N_{i,k}(u) M'_{j,l}(w), \tag{4.69}$$

$$\overline{D}_{uw} = \sum_{i=1}^{n+1}\sum_{j=1}^{m+1} h_{i,j} N'_{i,k}(u) M'_{j,l}(w), \tag{4.70}$$

$$\overline{D}_{uu} = \sum_{i=1}^{n+1}\sum_{j=1}^{m+1} h_{i,j} N''_{i,k}(u) M_{j,l}(w), \tag{4.71}$$

$$\overline{D}_{ww} = \sum_{i=1}^{n+1}\sum_{j=1}^{m+1} h_{i,j} N_{i,k}(u) M''_{j,l}(w). \tag{4.72}$$

The terms $N'_{i,k}(u)$, $N''_{i,k}(u)$, $M'_{j,l}(w)$ and $M''_{j,l}(w)$ are given by the equation 4.35.

NURBS surface will be used as geometry description for the NURBS-based thin-plate element, and with it will be necessary to compute its basis derivatives. As shown in thin plate elements there is the presence of second derivatives in the strain components which indicates that a C^1 continuity through the shape functions is mandatory. NURBS-based thin plate uses NURBS basis as shape functions, as it will be shown further. Above are demonstrated the surface derivatives, however it is its basis first and second derivatives that will come to use when computing the stiffness matrix, as it can be seen in the algorithm presented in the appendix A.3. To do so lets recall the NURBS surface basis Equations 4.55 and 4.56,

$$\begin{aligned} S_{i,j}(u,w) =& \frac{h_{i,j} N_{i,k}(u) M_{j,l}(w)}{\sum_{i=1}^{n+1}\sum_{j=1}^{m+1} h_{i1,j1} N_{i1,k}(u) M_{j1,l}(w)} = \frac{h_{i,j} N_{i,k}(u) M_{j,l}(w)}{Sum(u,w)}, \\ Sum(u,w) =& \sum_{i=1}^{n+1}\sum_{j=1}^{m+1} h_{i1,j1} N_{i1,k}(u) M_{j1,l}(w). \end{aligned}$$

The derivatives are obtained through formal differentiation, the first derivatives with respect to u and v respectively, can be done as

$$\begin{aligned} \frac{\partial R_{i,j}(u,w)}{\partial u} =& \frac{h_{i,j}\left(\frac{\partial}{\partial u} N_{i,k}(u)\right) M_{j,l}(w)}{Sum(u,w)} - \frac{h_{i,j} N_{i,k}(u) M_{j,l}(w)\left(\frac{\partial}{\partial u} Sum(u,w)\right)}{Sum(u,w)^2} \\ \frac{\partial R_{i,j}(u,w)}{\partial w} =& \frac{h_{i,j} N_{i,k}(u)\left(\frac{\partial}{\partial w} M_{j,l}(w)\right)}{Sum(u,w)} - \frac{h_{i,j} N_{i,k}(u) M_{j,l}(w)\left(\frac{\partial}{\partial w} Sum(u,w)\right)}{Sum(u,w)^2} \end{aligned} \tag{4.73}$$

It's second derivatives can be calculated by,

$$\begin{aligned}
\frac{\partial^2 R_{i,j}(u,w)}{\partial u^2} =& \frac{h_{i,j}\left(\frac{\partial^2}{\partial u^2}N_{i,k}(u)\right)M_{j,l}(w)}{Sum(u,w)} - \frac{2h_{i,j}\left(\frac{\partial}{\partial u}N_{i,k}(u)\right)M_{j,l}(w)\left(\frac{\partial}{\partial u}Sum(u,w)\right)}{Sum(u,w)^2} + \\
&+ \frac{2h_{i,j}N_{i,k}(u)M_{j,l}(w)\left(\frac{\partial}{\partial u}Sum(u,w)\right)^2}{Sum(u,w)^3} - \frac{h_{i,j}N_{i,k}(u)M_{j,l}(w)\left(\frac{\partial^2}{\partial u^2}Sum(u,w)\right)}{Sum(u,w)^2} \\
\frac{\partial^2 R_{i,j}(u,w)}{\partial w^2} =& \frac{h_{i,j}N_{i,k}(u)\left(\frac{\partial^2}{\partial w^2}M_{j,l}(w)\right)}{Sum(u,w)} - \frac{2h_{i,j}N_{i,k}(w)\left(\frac{\partial}{\partial w}M_{j,l}(w)\right)\left(\frac{\partial}{\partial w}Sum(u,w)\right)}{Sum(u,w)^2} + \\
&+ \frac{2h_{i,j}N_{i,k}(u)M_{j,l}(w)\left(\frac{\partial}{\partial w}Sum(u,w)\right)^2}{Sum(u,w)^3} - \frac{h_{i,j}N_{i,k}(u)M_{j,l}(w)\left(\frac{\partial^2}{\partial w^2}Sum(u,w)\right)}{Sum(u,w)^2} \\
\frac{\partial^2 R_{i,j}(u,w)}{\partial uw} =& \frac{h_{i,j}\left(\frac{\partial}{\partial u}N_{i,k}(u)\right)\left(\frac{\partial}{\partial w}M_{j,l}(w)\right)}{Sum(u,w)} - \frac{h_{i,j}\left(\frac{\partial}{\partial u}N_{i,k}(u)\right)M_{j,l}(w)\left(\frac{\partial}{\partial w}Sum(u,w)\right)}{Sum(u,w)^2} - \\
&- \frac{h_{i,j}N_{i,k}(w)\left(\frac{\partial}{\partial w}M_{j,l}(w)\right)\left(\frac{\partial}{\partial w}Sum(u,w)\right)}{Sum(u,w)^2} + \frac{h_{i,j}N_{i,k}(u)M_{j,l}(w)\left(\frac{\partial^2}{\partial uw}Sum(u,w)\right)}{Sum(u,w)^2} + \\
&+ \frac{2h_{i,j}N_{i,k}(u)M_{j,l}(w)\left(\frac{\partial}{\partial u}Sum(u,w)\right)\left(\frac{\partial}{\partial w}Sum(u,w)\right)}{Sum(u,w)^3}
\end{aligned} \tag{4.74}$$

given that the derivatives of $Sum(u,w)$ are equal to $\overline{D}$ showed above it is easy to compute the basis derivatives at a given point. The algorithm with a possible way is given in the appendix A.3.

Intentionally left blank.

Chapter 5

Isogeometric Analysis

In this chapter, isogeometric analysis will be presented. Concepts from previous chapters will be recalled and combined, as CAD geometry description and finite element analysis will feature in a close relation on this method. The process of meshing can be omitted, as the functions used in NURBS geometry description are adopted by the analysis for the geometry and for the solution field, whereas for the analysis nomenclature it will be considered as equal to the finite element method.

Before going through the method it is important to clarify that this chapter will strictly cover two dimension case, since it offers advantages in perception and eases the transformation for the Kirchhoff classic thin plate theory.

5.1 Isogeometric Analysis basic idea

The main concept for IGA is that basis functions used in CAD geometry description can work as shape functions for analysis, given that they fulfil the requirements, such as partition of unity and linear independence. Lets focus on Non-Uniform Rational B-Splines (NURBS) as they are the less limited geometry characterization, also the the most widespread in CAD technology.

Again, as shown in chapter 4 a NURBS patch is defined over parametric domain, which is divided into intervals by knot vectors. This intervals are defined as elements. In between knots (intervals) the B-spline basis functions gains form of polynomial which allows the use of Gauss quadrature on element level. As for NURBS basis in same intervals, take the form of rational polynomials, therefore the Gauss quadrature with NURBS is only approximative. In this work, standard Gauss-Legendre quadrature will be implemented, although being only approximative it has been proved to be reliable [1]. In [33, 34], an optimal quadrature rule – the half-point rule – for IGA was presented.

This NURBS elements, equivalently to finite elements in FEM, are also defined by nodes and each node have a corresponding basis function. The nodes are what was called, in chapter 4, as control points where degrees of freedom and boundary conditions are applied. Another important characteristic about IGA is that basis functions are spread along series of elements and not just one as in finite element method as represented in Figure 5.1. It allows higher continuity shape functions over element boundaries and it is defined by the order of the patch, the higher the order the higher the continuity of the basis functions and, while it increases solution accuracy it causes more dependency between elements. Given this for orders bigger than the second, the term element is

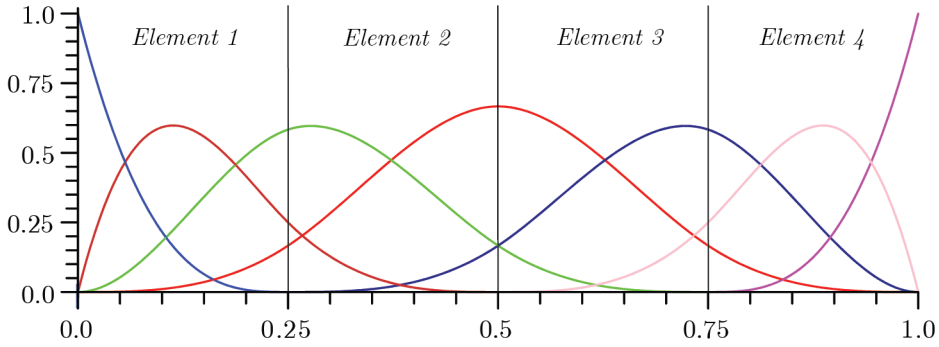


Figure 5.1: Representation of basis functions extending over other elements (adapted from [14]).

"controversial" since it is not independent elementary parts that can be put to build the whole model, but for analysis this elements can be treated exactly as in the classical finite element theory with the difference that the shape functions are the basis that define the geometry. In the Table 5.1 is an a summary of the similarities and differences of FEM and IGA.

Table 5.1: Comparison between finite element analysis and isogeometric analys with NURBS basis [1].

Finite element method	Similarities	Isogeometric analysis
Nodal points		Control points
Nodal variables		Control variables
Mesh		Knots
Basis interpolates nodal points and variables		Basis does not interpolate control points and variables
Approximate geometry		Exact geometry
Polynomial basis		NURBS basis
Subdomains		Patches
Gibbs phenomena		Variation diminishing
	Compact support	
	Partition of unity	
	Isoparametric concept	
	Affine covariance	
	Patch tests satisfied	

Before going on with the method formulation it is important to summarize the properties of NURBS basis for analysis:

1. Linear independence and partition of unity;
2. degrees of freedom are applied in control points;
3. Basis have higher-order continuities through the element boundaries;
4. Isoparametric concept is used;

5. The basis which capture the geometry is used to approximate the unknown field.

5.2 Isogeometric discretization

NURBS mappings transform coordinates from parameter space $\hat{\Omega}$ to physical domain which is denoted by Ω . The mapping from parametric domain into the physical space is given by,

$$\mathbf{x} = \sum_{I=1}^{n} \Phi_I(\xi)\mathbf{B}_I, \tag{5.1}$$

where Φ represents the shape functions. Shape function can either come form the univariate NURBS basis if Ω is a curve or the bivariate NURBS basis functions if it is a surface. In two dimensions the parametric domain is a square. B_I indicates the control point number I. For last ξ represents the parametric coordinate in two dimensions it is determined as $\xi,\ \eta$.

For isoparametric formulation, the same shape functions approximate the unknown field, and it is determined as,

$$\mathbf{u}(x) = \sum_{I=1}^{n} \Phi(\xi)\mathbf{u}_I, \tag{5.2}$$

where $\mathbf{u}_I$ denotes the value of the field variable at a control point I. Here I is referred as a control variable or degree of freedom (d.o.f.). It is important to realise that that control points are not always located inside the domain of interest, meaning that $\mathbf{u}_I$ can be outside of the domain, and does not represent a physical nodal value as conventional FEM. Also for control points inside the domain due to non-interpolatory nature of NURBS.

To determine the spatial derivatives of the shape funtion, it is necessary to obtain the Jacobian matrix in order to map the derivatives from the parametric domain over to the physical domain. This geometry mapping is defined by the following matrix,

$$\mathbf{J}_\xi = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix}, \tag{5.3}$$

where the components are determined as Equation 5.1.

$$\frac{\partial x_i}{\partial \xi_j} = \frac{\partial \Phi}{\partial \xi_j} B_{iI}, \tag{5.4}$$

B_{iI} represents the *ith* coordinate of a certain control point I. The determinant of the Jacobian matrix is represented as $|\mathrm{J}_\xi|$. The derivatives with respect to the physical domain can now be calculated as shown in the following transformation,

$$\begin{Bmatrix} \frac{\partial \Phi_I}{\partial \xi} \\ \frac{\partial \Phi_I}{\partial \eta} \end{Bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix} \begin{Bmatrix} \frac{\partial \Phi_I}{\partial x} \\ \frac{\partial \Phi_I}{\partial y} \end{Bmatrix} \quad ; \quad \begin{Bmatrix} \frac{\partial \Phi_I}{\partial x} \\ \frac{\partial \Phi_I}{\partial y} \end{Bmatrix} = \mathbf{J}_\xi^{-1} \begin{Bmatrix} \frac{\partial \Phi_I}{\partial \xi} \\ \frac{\partial \Phi_I}{\partial \eta}. \end{Bmatrix} \tag{5.5}$$

5.3 Numerical integration

Before going through the with the numerical integration, it is imperative for analysis to be performed, the addition of an additional space which is called *parent space* $\square(\overline{\xi}, \overline{\eta})$. As represented in Figure 5.2 this is the space where integration rules are imposed since parent domain is defined over $\square = [-1, 1]$. With an additional domain comes an additional mapping, as seen in the figure there is a parent to parametric and a parametric to physical domain mapping.

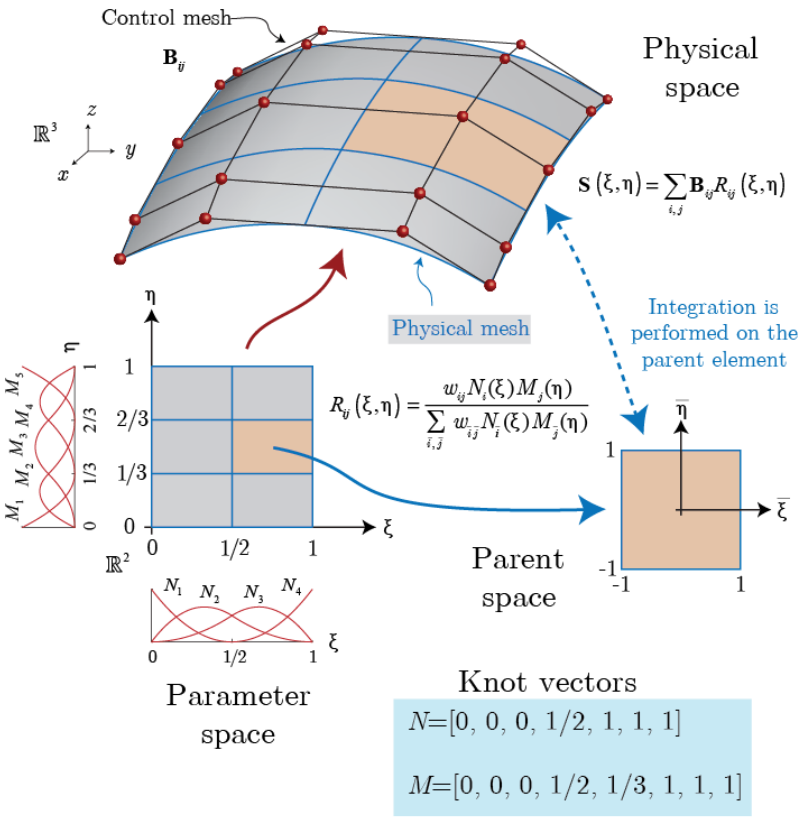


Figure 5.2: Representation of mapping spaces (adapted from [34]).

A two dimensional element defined as $\hat{\Omega} = [\xi_i,\ \xi_{i+1}] \times [\eta_j,\ \eta_{j+1}]$ is mapped from parent to parametric space through

$$\xi = \frac{1}{2}[(\xi_{i+1} - \xi_i)\overline{\xi} + (\xi_{i+1} + \xi_i)], \tag{5.6}$$

$$\eta = \frac{1}{2}[(\eta_{j+1} - \eta_j)\overline{\eta} + (\eta_{j+1} + \eta_j)], \tag{5.7}$$

where $\overline{\xi}$ and $\overline{\eta}$ are Gauss quadrature points. The determinant of the Jacobian of this

transformation is given by,

$$| \mathbf{J}_{\overline{\xi}}| = \frac{1}{4}(\xi_{i+1} - \xi_i)(\eta_{j+1} + \eta_j) \tag{5.8}$$

The entire physical domain is split into element integrals over an element domain denoted as Ω_e. This integrals are picked from the parametric domain $\hat{\Omega}_e$ which is pulled back from the parent domain $\square$ where the integration rules are defined. Considering a function f with two variables $(x,\, y)$,

$$\begin{aligned}\int_{\mathbb{R}} f(x,y)\mathrm{d}\Omega &= \sum_{e=1}^{n} \int_{\Omega_e} f(x,y)\ \mathrm{d}\Omega = \sum_{e=1}^{n} \int_{\hat{\Omega}_e} f(x(\xi), y(\eta))|J_\xi|\ \mathrm{d}\Omega_e \\ &= \sum_{e=1}^{n} \int_{\square} f(\overline{\xi}, \overline{\eta})|\mathbf{J}_\xi||\mathbf{J}_{\overline{\xi}}|\ \mathrm{d}\square\end{aligned} \tag{5.9}$$

The final integral can be evaluated using standard Gauss quadrature. It is recommended a $(k+1) \times (l+1)$ quadrature, for two dimensional elements, where k and l denote the orders in each direction ξ and η of NURBS-basis, respectively.

When implementing, it is normally necessary another space known as index space. It does not affect formulation it is only a coordinate vector that specifies each knot with a distinct coordinate values, the aim is then to discard elements of non-zero parametric, a example of this space is illustrated in Figure 5.3, with the same knot vectors from the Figure 5.2.

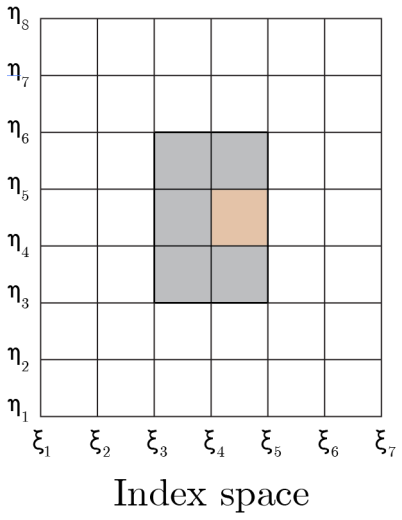


Figure 5.3: Index space (adapted from [34]).

5.4 Assembly for two dimensional analysis

Known how the discretization and integration are done with this method, it is now important to comprehend the assembly process for two dimensional analysis, hence, it

has more similarities with thin plate analysis. Recalling the virtual work done by forces equation (2.30) from Chapter 2,

$$\iint_{A^{(e)}} \delta\varepsilon^{\mathrm{T}}\sigma t \ \mathrm{d}A - \iint_{A^{(e)}} \delta\mathbf{u}^{\mathrm{T}}\mathbf{b}t \ \mathrm{d}A - \oint_{l^{(e)}} \delta\mathbf{u}^{\mathrm{T}}\mathbf{t}t \ \mathrm{d}s = [\delta a^{(e)}]^{\mathrm{T}}\mathrm{q}^{(e)}, \tag{5.10}$$

the virtual displacement δu is defined with NURBS-basis,

$$\delta\mathrm{u} = \Phi\delta a^{(e)} \ \ ; \ \ \delta\varepsilon = \mathbf{B}\delta\mathrm{a}^{(e)}, \tag{5.11}$$

replacing these into the equation (5.10), and considering the arbitrary of virtual displacements, comes,

$$\iint_{A^{(e)}} \mathbf{B}^{\mathrm{T}}\sigma t \ \mathrm{d}A - \iint_{A^{(e)}} \Phi^{\mathrm{T}}\mathbf{b}t \ \mathrm{d}A - \oint_{l^{(e)}} \Phi^{\mathrm{T}}\mathbf{t}t \ \mathrm{d}s = q^{(e)}, \tag{5.12}$$

that can be expressed in matrix form as,

$$\mathbf{K}^{(e)}\mathbf{a}^{(e)} - \mathbf{f}^{(e)} = \mathbf{q}^{(e)}, \tag{5.13}$$

where the stiffness matrix $\mathbf{K}$ is obtained as,

$$\mathbf{K}^{(e)} = \iint_{A^{(e)}} \mathbf{B}^{T}\mathbf{D}\mathbf{B}t \ \mathrm{d}A \tag{5.14}$$

the matrix $\mathbf{B}$, in two dimensions is given by

$$\mathbf{B} = \begin{bmatrix} \frac{\partial\Phi_I}{\partial x} & 0 \\ 0 & \frac{\partial\Phi_I}{\partial y} \\ \frac{\partial\Phi_I}{\partial y} & \frac{\partial\Phi_I}{\partial x} \end{bmatrix} \tag{5.15}$$

where the derivatives are computed according to the equation (5.5).

5.5 Mesh refinement

Refinement leads to better accuracy,however it is important in the analysis to carefully define the refinement parameters, since with more refinement, normally, comes bigger processing times. There are 3 types of refinement for IGA, in which two are analogous to FEM refinement: knot insertion (an analogue to h-refinement), the order-elevation (an analogue to p-refinement) and a new alternative, that takes advantage of the fact that knot insertion and order elevation do not commute, called "k-refinement".

5.5.1 Knot insertion

Knots can be inserted without changing the curve geometrically or parametrically. Given a knot vector $X = [x_1, x_2, ..., x_{n+k+1}]$ and $\overline{X} = [x_p, x_{k+1}]$ the desired new knot. The new basis functions are obtained recursively with Equations 4.32 and 4.33 with the new

knot vector $Y = [y_1, y_2, ..., y_{m+k+1}]$ where $\overline{X} \in Y$ and $m > n$. The new $n+1$ control points $\overline{B}$ vector is formed from the initial one B_i by using the Oslo algorithm [3,35,36],

$$\overline{B_i} = \sum_{i=1}^{n+1} \alpha_{i,j}^k B_i \quad 1 \leq i \leq n, \quad 1 \leq j \leq m, \tag{5.16}$$

where the $a_{i,j}^k$ are given by the recursion relation,

$$a_{i,j}^1 = \begin{cases} 1 & x_i \leq y_j \leq x_{i+1} \\ 0 & \text{otherwise} \end{cases}, \tag{5.17}$$

$$a_{i,j}^k = \frac{y_{j+k-1} - x_i}{x_{i+k-1} - x_i} a_{i,j}^{k-1} + \frac{x_{i+k} - y_{j+k-1}}{x_{i+k} - x_{i+1}} a_{i+1,j}^{k-1}. \tag{5.18}$$

After the insertion of a knot value if the original knot vector was uniform, then, it can became nonuniform [3]. However, this can be overcome by inserting multiple knot values midway in each existing non-zero interval. Continuity of the curve may be preserved if each unique internal knot value appear no more than k times.

Figure 5.4 shows an interior knot insertion of $\overline{x} = 0.5$. It can be seen in the figure that the new curve is defined by four control points instead of three, although the curve did not change geometrically nor parametrically. The mesh was partitioned and the new basis is more richer as there are one more element and one more basis.

5.5.2 Order elevation

As well as for knot insertion, the elevation of polynomial order of basis functions may be increased without changing the geometry or parametrization. Also, when polynomials are order elevated, each unique knot must be repeated to preserve continuity in the *kth* derivative of the curve being elevated [1]. Order elevation offers advantages for differentiation as the spline remains infinitely differentiable, whereas for knot insertion there is reduced differentiability at the inserted knots. Depending on the knot multiplicity of the existing knots there is a number of new knots.

Piegl and Tiller have taken an implementation approach where first the NURBS curve is decomposed into Bézier curve segments by inserting $k-1$ knots values at each knot. These segments are order elevated. In the end, unnecessary knots are removed. The details of the algorithm can be found in [2, 37].

A general solution was presented in [38] for B-spline order elevation. This approach have somewhat complex mathematical details for sake of brevity they are avoided. Restricting to fundamentals, when a B-spline curve is order elevated the new curve must be identical to the original one,

$$P(t) = \sum_{i=1}^{n+1} B_i N_{i,k} = \sum_{i=1}^{m+1} \overline{B}_i N_{i,k+1} \quad m+1 > n+1, \tag{5.19}$$

where $\overline{B}$ are the new control polygon vertices for the new curve. The original knot vector is denoted as

$$X = [\underbrace{0, \ldots, 0}_{k}, \underbrace{a, \ldots, a}_{p_1}, \ldots, \underbrace{b, \ldots, b}_{p_s}, \ldots, \underbrace{n-k+2, \ldots, n-k+2}_{k}], \tag{5.20}$$

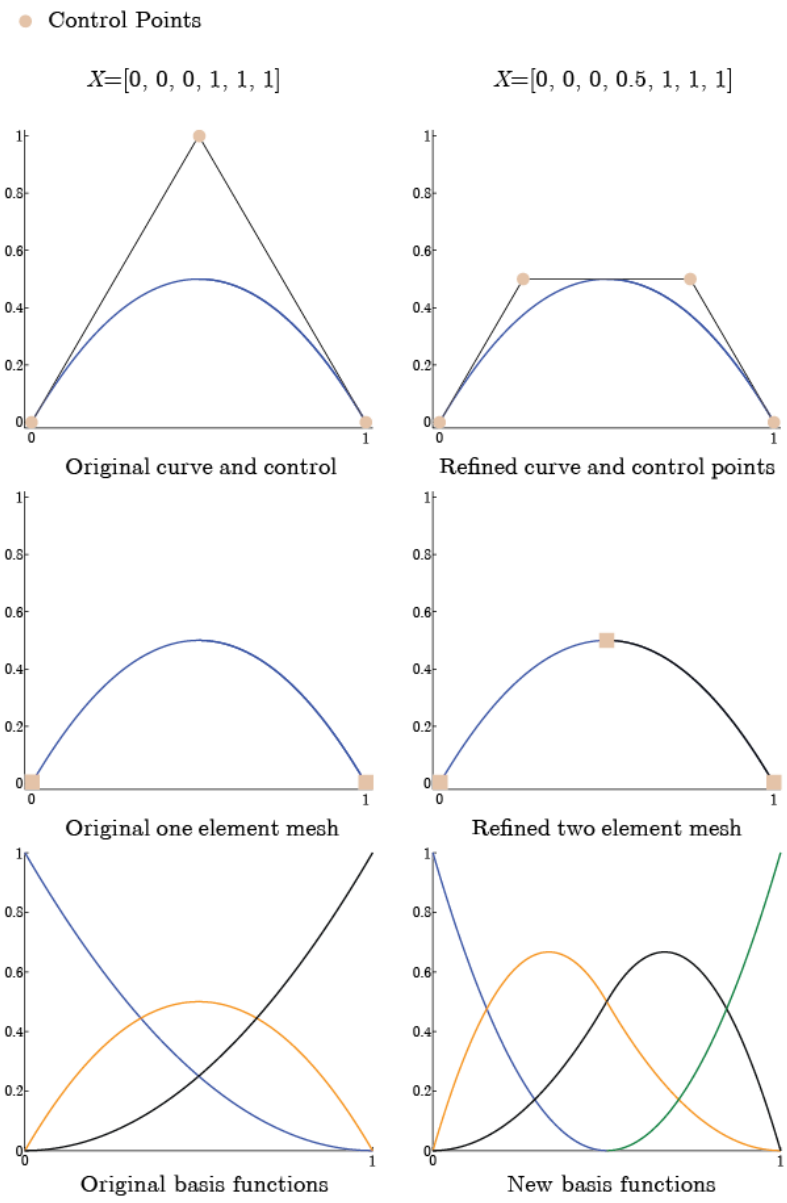


Figure 5.4: Knot insertion refinement method (adapted from [34]).

where p_i defines the multiplicity of any internal knot values and s measures the number of occurrences of multiple internal knot values.

Recalling the effect of multiplicity of internal knots on B-spline basis functions from section 4.3.2, it decreases continuity resulting in a curve closer to the control polygon vertices. Considering that $P(t)$ must have the same continuity at each internal knot after the order elevation, the new knot vector has multiplicity $k+1$ at the end knots and $m+1$ for internal knot values. The resulting knot vector is,

$$Y = [\underbrace{0,\ldots,0}_{k+1},\underbrace{a,\ldots,a}_{p_1+1},\ldots,\underbrace{b,\ldots,b}_{p_s+1},\ldots,\underbrace{n-k+2,\ldots,n-k+2}_{k+1}], \tag{5.21}$$

whereas for the number of the control point in the new control polygon, $\overline{B}$, is $\overline{n} = n + s + k + 1$.

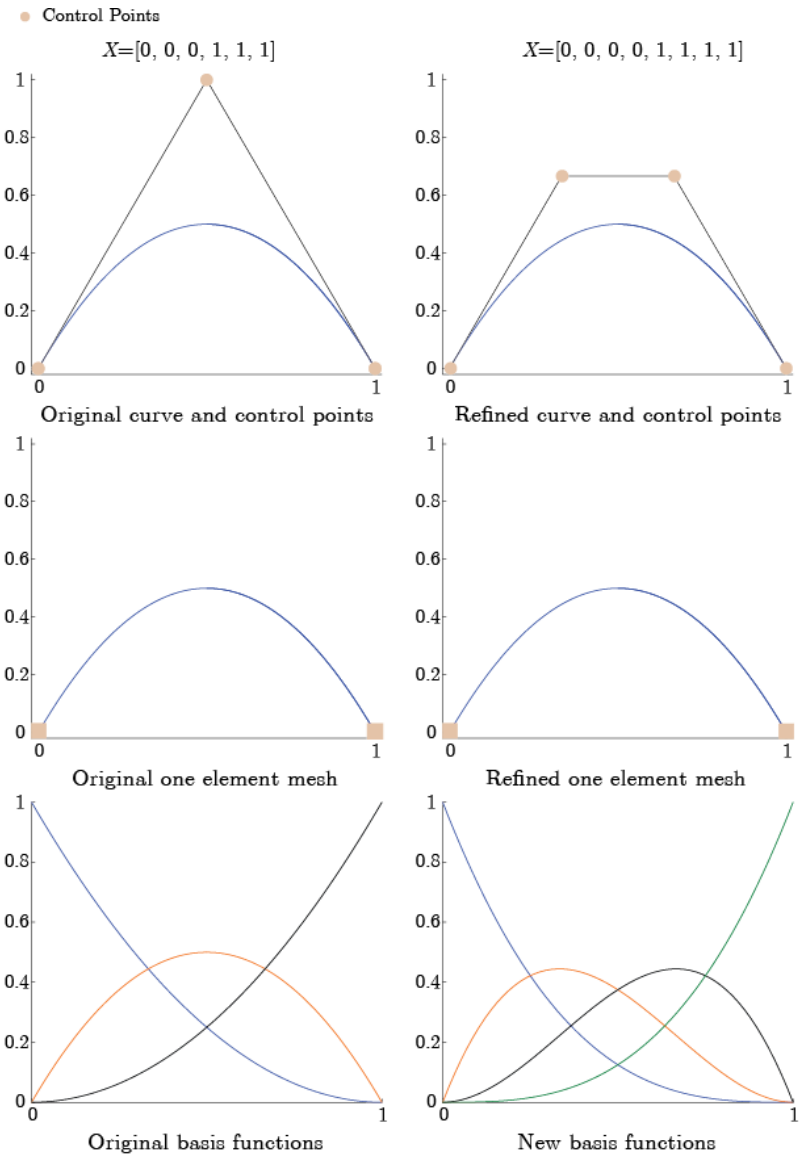


Figure 5.5: Order elevation refinement method (adapted from [34]).

Figure 5.5 shows a order elevation example, where the original curve has order $k = 2$ and the refined one $k = 3$. It can be seen that the multiplicity of end knots elevates by one. Control points are changed where there is one more control point defining the curve, although the curve does not change nor geometrically nor parametrically, contrary to knot insertion, the refined curve mesh maintains the original number of elements, note that the position of the control points are different then with knot insertion. The basis functions of the new curve, in comparison to the original one, is richer as there is one more basis.

5.5.3 k-Refinement

In [1] an alternative for order elevation was presented referred above as k-refinement. Consider a new knot $\overline{x}$ which is inserted in a knot vector of a kth order curve, the number of continuous derivatives of the basis function at the new knot is $k-1$. If then order elevation is implemented to the new curve, raising it to a new $\overline{k}th$ order, every distinct knot including the inserted one, increases multiplicity, resulting in a preservation of the discontinuities in the kth derivative of the basis function so, basis still has $k-1$ continuous derivatives at $\overline{x}$. Taken this into account, and if instead the order elevation is done first to a the same $\overline{p}th$ order and then the desired unique knot insertion $\overline{x}$, the basis have then $\overline{q}-1$ continuous derivatives at $\overline{x}$, this is the k-refinement procedure.

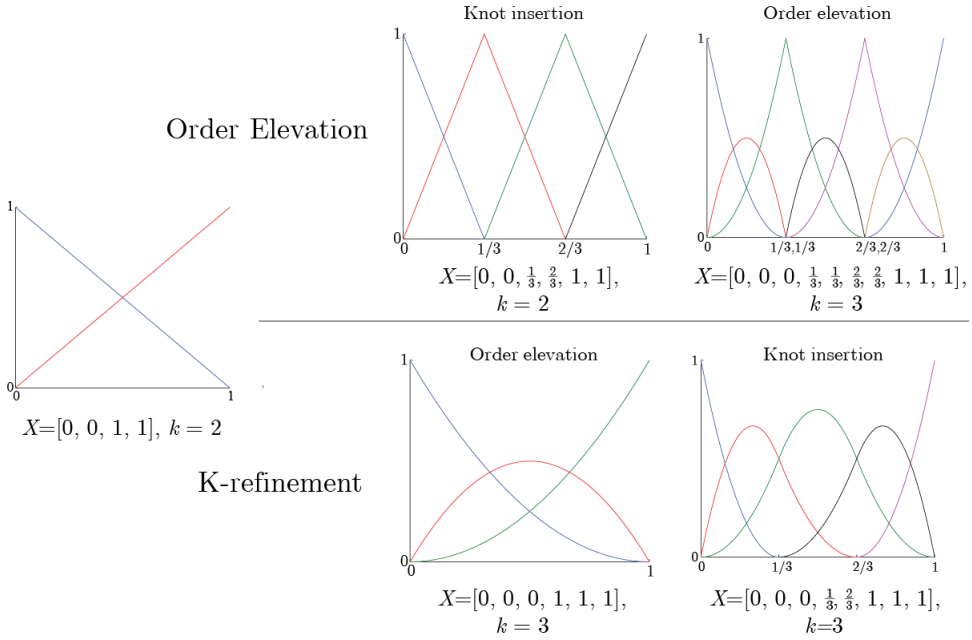


Figure 5.6: Comparison between order elevation and k-refinement (adapted from [34]).

Figure 5.6 makes a comparison between order elevation and k-refinement, both refinements done from the same basis. It can be seen that the end basis for k-refinement has continuity of C^{k-1} across boundaries, where order elevation has C^0 continuity as it maintains continuity across boundaries. There is also a bigger number of basis in order elevation, as already mentioned, elevating order by one elevates multiplicity in all knots by one. This does not happen in k-refinement as firstly the order elevation is done and only then the knot insertion, which means that internal knot does not increase multiplicity.

5.6 Code architecture

In this section, both FEM and IGA implementation structures will be analysed It is important to understand that the presented code structure is divided in three parts: pre-processing, processing and post-processing. The structure for IGA can be very similar to FEM, it differs in the shape functions which are defined by spline basis functions, and isogeometric analysis formulation makes use of the parent space $\square(\overline{\xi}, \overline{\eta})$ and index space. Through the section FEM and IGA structures will be discussed in detail. Figure 5.7 demonstrates a flowchart of a simple FEM code, the coloured routines are the ones that differs from IGA and will be commented in the following.

Pre-Processing is where known variables are set and defined. These variables are normally:

1. Material properties (i.e. Young modulus and Poisson coefficient);
2. Geometry definition (Coordinates);
3. Number of degrees of freedom;
4. Pressure/ load to be applied over the geometry;
5. The Gauss quadrature.

With these introduced there are some routines that help to set all variables that are needed to proceed to processing stage. If the geometry definition only defines nodes and not elements, some routine has to generate and define the mesh type and connectivity. Also material matrices can be sort in this part of the code, as example the constitutive matrix, if constant through the elements.

From Figure 5.7 it is possible to notice two routines that differ. The input data change as the file format will depend on the technology being used. The precise forms of connectivity and global matrices depend on the basis.

The mesh connectivity can differ, for FEM an element is defined by the nodes at its boundary, if it is a 4-node element it will have 4 nodes indexed to each element in the mesh connectivity matrix. IGA is not so linear, the element connectivity depends on the spline order as $element = [noElements, k \times l]$ as example, the element of a $3th$ order surface ($k = 3$ and $l = 3$) is defined by 9 control points. Only for the first order spline, will the element be equivalent to FEM.

There is also the need to set the index space again, where non-zero intervals of the knot vector are indexed, and a matrix with the element range which is carries all knot intervals.

Processing is the when the code loops through elements in order to add the contributions of all nodal variables. After everything is set in pre-processing stage, the global stiffness matrix as well as the force vector are put to zero. The stiffness matrix and force vector algorithms can be find in the appendices A.2 for FEM and A.3 for IGA formulations.

Regarding FEM process, there is a loop through the elements, and within a loop through the Gaussian integration points. In each quadrature point the shape functions are evaluated as its derivatives, there is a transformation from the parametric space to physical space then, the node contributions are added to the element stiffness matrix.

Ending the loop, all element stiffness matrices are assembled resulting in the global stiffness matrix.

Considering IGA process is considerably different, there is also a loop through the elements and within a loop through the integration points. Recalling the index space, it is important to comprehend that each element carries index which is then used to extract the respective element range as shown in the algorithm presented in appendix A.3. When looping through the integration points there is a transformation from parent to parametric space. It is in this latter space that the basis functions and its derivatives are evaluated and then there is a last transformation from parametric to physical space, the contributions are added to the element stiffness matrix. Ending the loop all element matrices are assembled to build the global stiffness matrix.

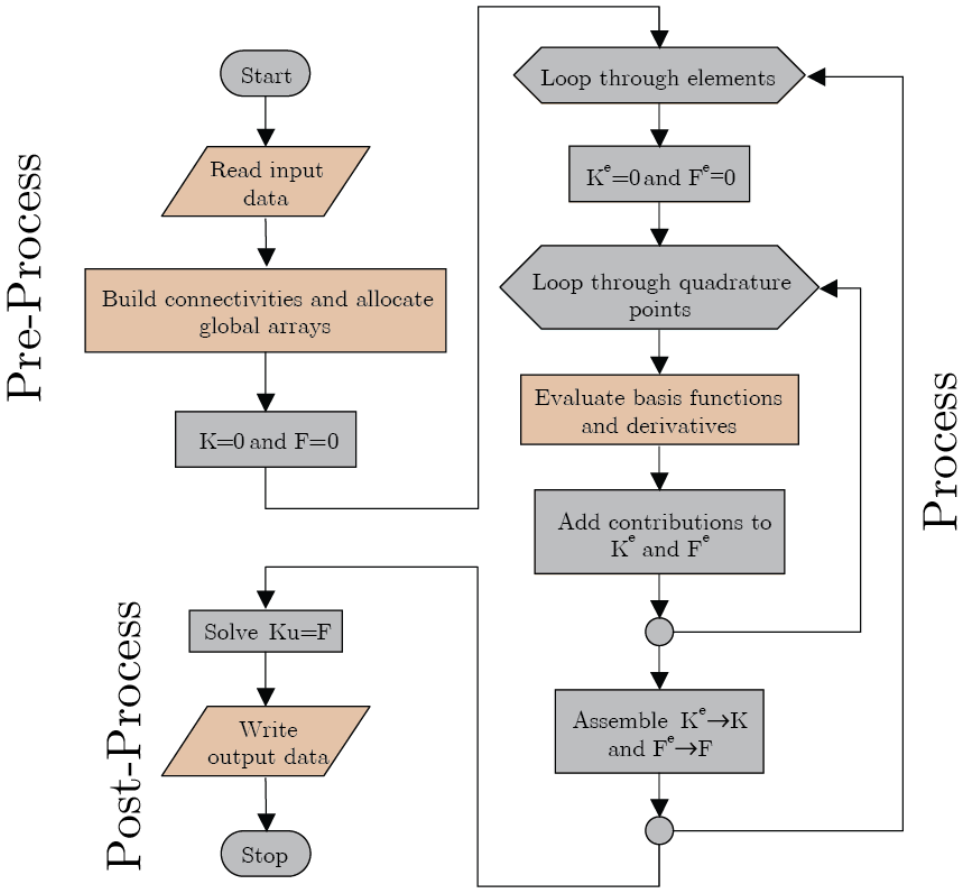


Figure 5.7: Basic implementation structure of FEM. A IGA representation differs in the coloured routines.

Post-Processing where the nodal displacement is evaluated. The user introduces the necessary boundary conditions and also local forces can be applied. These are then multiplied by the d.o.f.'s and the result is a position vector of all nodes.

As shown in the flowchart, solving the equation $\mathbf{Ku} = \mathbf{F}$, leads to the output values. In the FEM case, these are in the physical geometry, whereas for IGA the output of the

displacement **u** is at the control points which are not in the physical domain, to do so it is needed to write in a NURBS format, so the resulting physical geometry is specified and the physical displacement determined.

Intentionally left blank.

Chapter 6

NURBS-based Kirchhoff thin plate element

For isogeometric thin plate analysis, the NURBS plate, designed in CAD, can be used as a model for analysis. For a typical FEM formulation, the use of rotations degrees of freedom, implies that this directors have to be assigned to the geometry in the analysis model, since they don't exist in the design model. The goal of IGA , as mentioned before is to use the exact geometry description, and try to shorten the distance between design and analysis. Rotation-free thin plate formulation is an example of formulation that allows the use of the same model, for design and analysis [14]. Hence, the following formulation will be rotation-free.

6.1 NURBS-based thin plate formulation

The deflection w is defined with NURBS basis as follows,

$$w = R_i(\xi, \eta) w_i, \tag{6.1}$$

where $R_i(\xi, \eta)$ is NURBS basis functions associated with node i, w_i is the nodal displacement.

Recalling element stiffness matrix defined as equation (3.47) in chapter 3,

$$\mathbf{K}_{ij}^{(e)} = \int\int_{A^{(e)}} \mathbf{B}_{bi}^{\mathrm{T}} \hat{\mathbf{D}}_b \mathbf{B}_{bj} \, \mathrm{d}x\mathrm{d}y,$$

where the generalized constitutive matrix $\hat{\mathbf{D}}_b$ reads,

$$\hat{\mathbf{D}}_b = \frac{Et^3}{12(1-\nu^2)} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix}.$$

The element bending strain matrix $\mathbf{B}$ is reduced to only the first degree of freedom, which is given by the first column of the Matrix 3.44, where the shape functions are now

given by the NURBS-basis is defined as

$$\mathbf{B}^{(e)} = \begin{bmatrix} \frac{\partial^2 R_1}{\partial x^2} & \frac{\partial^2 R_2}{\partial x^2} & \cdots & \frac{\partial^2 R_n}{\partial x^2} \\ \frac{\partial^2 R_1}{\partial y^2} & \frac{\partial^2 R_2}{\partial y^2} & \cdots & \frac{\partial^2 R_n}{\partial y^2} \\ 2\frac{\partial^2 R_1}{\partial xy} & 2\frac{\partial^2 R_2}{\partial xy} & \cdots & 2\frac{\partial^2 R_n}{\partial xy} \end{bmatrix}, \tag{6.2}$$

where n denotes the indexed number denotes the number of basis functions of the element (e). The second derivatives of shape functions represented, are with respect to physical coordinates. The derivatives of shape functions with respect to natural coordinates are given by

$$\begin{aligned} \frac{\partial R}{\partial \xi} =& \frac{\partial R}{\partial x}\frac{\partial x}{\partial \xi} + \frac{\partial R}{\partial y}\frac{\partial y}{\partial \xi}, \\ \frac{\partial R}{\partial \eta} =& \frac{\partial R}{\partial x}\frac{\partial x}{\partial \eta} + \frac{\partial R}{\partial y}\frac{\partial y}{\partial \eta}, \end{aligned} \tag{6.3}$$

transforming to matrix form it comes,

$$\begin{bmatrix} \frac{\partial R}{\partial \xi} \\ \frac{\partial R}{\partial \eta} \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix} \begin{bmatrix} \frac{\partial R}{\partial x} \\ \frac{\partial R}{\partial y} \end{bmatrix}, \tag{6.4}$$

where the 2×2 matrix is the Jacobian $\mathbf{J}$. For a n node element, the shape functions are written as,

$$\begin{bmatrix} \frac{\partial R_1}{\partial x} & \frac{\partial R_2}{\partial x} & \cdots & \frac{\partial R_n}{\partial x} \\ \frac{\partial R_1}{\partial y} & \frac{\partial R_2}{\partial y} & \cdots & \frac{\partial R_n}{\partial y} \end{bmatrix} = \mathbf{J}^{-1} \begin{bmatrix} \frac{\partial R_1}{\partial \xi} & \frac{\partial R_2}{\partial \xi} & \cdots & \frac{\partial R_n}{\partial \xi} \\ \frac{\partial R_1}{\partial \eta} & \frac{\partial R_2}{\partial \eta} & \cdots & \frac{\partial R_n}{\partial \eta} \end{bmatrix}, \tag{6.5}$$

Proceeding to second derivatives of shape functions,

$$\begin{aligned} \frac{\partial^2 R}{\partial \xi^2} =& \frac{\partial}{\partial \xi}\left(\frac{\partial}{\partial \xi}\right) = \frac{\partial}{\partial \xi}\left(\frac{\partial R}{\partial x}\frac{\partial x}{\partial \xi} + \frac{\partial R}{\partial y}\frac{\partial y}{\partial \xi}\right) \\ =& \frac{\partial R}{\partial x}\frac{\partial^2 x}{\partial \xi^2} + \frac{\partial x}{\partial \xi}\left(\frac{\partial^2 R}{\partial x^2}\frac{\partial x}{\partial \xi} + \frac{\partial^2 R}{\partial xy}\frac{\partial y}{\partial \xi}\right) + \frac{\partial R}{\partial y}\frac{\partial^2 x}{\partial \xi^2} + \frac{\partial y}{\partial \xi}\left(\frac{\partial^2 R}{\partial xy}\frac{\partial x}{\partial \xi} + \frac{\partial^2 R}{\partial y^2}\frac{\partial y}{\partial \xi}\right), \end{aligned} \tag{6.6}$$

and using the same process for the other natural coordinate η and for the cross-derivative comes,

$$\frac{\partial^2 R}{\partial \eta^2} = \frac{\partial R}{\partial x}\frac{\partial^2 x}{\partial \eta^2} + \frac{\partial x}{\partial \eta}\left(\frac{\partial^2 R}{\partial x^2}\frac{\partial x}{\partial \eta} + \frac{\partial^2 R}{\partial xy}\frac{\partial y}{\partial \eta}\right) + \frac{\partial R}{\partial y}\frac{\partial^2 x}{\partial \eta^2} + \frac{\partial y}{\partial \eta}\left(\frac{\partial^2 R}{\partial xy}\frac{\partial x}{\partial \eta} + \frac{\partial^2 R}{\partial y^2}\frac{\partial y}{\partial \eta}\right), \tag{6.7}$$

$$\frac{\partial^2 R}{\partial \xi \eta} = \frac{\partial R}{\partial x}\frac{\partial^2 x}{\partial \xi \eta} + \frac{\partial R}{\partial y}\frac{\partial^2 y}{\partial \xi \eta} + \frac{\partial x}{\partial \xi}\frac{\partial x}{\partial \eta}\frac{\partial^2 R}{\partial x^2} + \left(\frac{\partial x}{\partial \xi}\frac{\partial y}{\partial \eta} + \frac{\partial x}{\partial \eta}\frac{\partial y}{\partial \xi}\right)\frac{\partial^2 R}{\partial xy} + \frac{\partial y}{\partial \xi}\frac{\partial y}{\partial \eta}\frac{\partial^2 R}{\partial y^2}. \tag{6.8}$$

In matrix form, the previous equations can be written as

$$\begin{bmatrix} \frac{\partial^2 R}{\partial \xi^2} \\ \frac{\partial^2 R}{\partial \eta^2} \\ \frac{\partial^2 R}{\partial \xi \eta} \end{bmatrix} = \begin{bmatrix} \frac{\partial^2 x}{\partial \xi^2} & \frac{\partial^2 y}{\partial \xi^2} & 2\frac{\partial x}{\partial \xi}\frac{\partial y}{\partial \xi} \\ \frac{\partial^2 x}{\partial \eta^2} & \frac{\partial^2 y}{\partial \eta^2} & 2\frac{\partial x}{\partial \eta}\frac{\partial y}{\partial \eta} \\ \frac{\partial x}{\partial \eta}\frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi}\frac{\partial y}{\partial \eta} & \frac{\partial x}{\partial \xi}\frac{\partial y}{\partial \eta} + \frac{\partial x}{\partial \eta}\frac{\partial y}{\partial \xi} \end{bmatrix} \begin{bmatrix} \frac{\partial^2 R}{\partial x^2} \\ \frac{\partial^2 R}{\partial y^2} \\ \frac{\partial^2 R}{\partial xy} \end{bmatrix} + \begin{bmatrix} \frac{\partial^2 x}{\partial \xi^2} & \frac{\partial^2 y}{\partial \xi^2} \\ \frac{\partial^2 x}{\partial \eta^2} & \frac{\partial^2 y}{\partial \eta^2} \\ \frac{\partial^2 x}{\partial \xi \eta} & \frac{\partial^2 y}{\partial \xi \eta} \end{bmatrix} \begin{bmatrix} \frac{\partial R}{\partial x} \\ \frac{\partial R}{\partial y} \end{bmatrix}, \tag{6.9}$$

which allows the computation of the second derivative with respect to physical coordinates, that comes as,

$$\begin{bmatrix} \frac{\partial^2 R}{\partial x^2} \\ \frac{\partial^2 R}{\partial y^2} \\ \frac{\partial^2 R}{\partial xy} \end{bmatrix} = \mathbf{J}_{33}^{-1} \left(\begin{bmatrix} \frac{\partial^2 R}{\partial \xi^2} \\ \frac{\partial^2 R}{\partial \eta^2} \\ \frac{\partial^2 R}{\partial \xi \eta} \end{bmatrix} - \mathbf{J}_{32}^{-1} \begin{bmatrix} \frac{\partial R}{\partial x} \\ \frac{\partial R}{\partial y} \end{bmatrix} \right). \tag{6.10}$$

6.2 Boundary conditions

Considering finite element formulation, the application of boundary conditions is really straight forward, if one needs to set some value to a node displacement or rotation, it is just a matter of setting the respective d.o.f. to the desired value. Considering isogeometric analysis the control points are where d.o.f.'s are applied, the non-interpolatory nature of NURBS can create some difficulties in imposing boundary conditions. It is known that for open knot vectors the boundary knots are interpolatory, which is very important given that the application of restrictions in that domain can be as trivial as FEM for homogeneous boundary conditions.

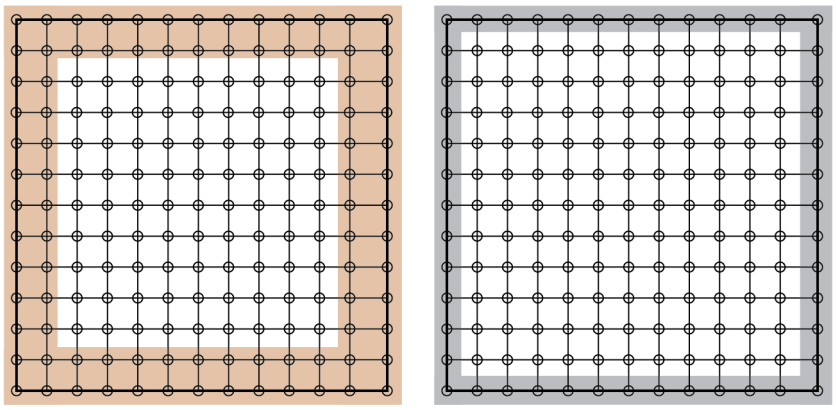


Figure 6.1: Representative images of boundary conditions. Figure a) shows a fixed boundary, where 2 rows are constrained to zero displacement. Figure b) is representative of a simply supported boundary where only one row is constrained at the boundary.

The studied NURBS-based thin plate element has only 1 dof per node, therefore it can become difficult to establish some restrictions. However there are two types of boundary conditions that can be applied: fixed support (or clamped edge) and simple support. A

simply supported edge, shows rotation contrary to the clamped edge that fixes rotations on the boundary nodes, note that having only displacement d.o.f.'s does not mean that rotations cannot be restricted. A simply supported edge has zero displacement $u = 0$ along itself. For a fixed edge, in order to fix rotations it is needed to set zero displacement within two row's of control points at the boundary.

The practical application of boundary conditions for IGA can be done through the following steps:

a) Create a vector with the index of restricted d.o.f.'s;

b) Setting the difference of this vector with the all d.o.f.'s it is created an active d.o.f. vector with all non constrained d.o.f.'s;

c) It is then possible to determine the displacement of these active d.o.f.'s using the ; $\mathbf{u}(activedof) = \mathbf{K}^{-1}(activedof, activedof)/\mathbf{f}(activedof)$

d) Then a final displacement vector $\mathbf{u}$ is created with the fixed d.o.f.'s.

Figure 6.1 is a representation of a square plate with fixed (in red) and simply supported (in blue) boundaries.

Chapter 7

Numerical examples

In this chapter, the developed isogeometric analysis and finite element method codes, are going to be tested through a series of benchmark examples. The NURBS-based thin plate element will be evaluated with three distinct orders $k = 3$, $k = 4$ and $k = 5$, where k-refinement is applied which means firstly the order is elevated and then the mesh refinement. The solutions will be compared with other elements such as MCZ thin plate element with three degrees of freedom per node, as well as Abaqus commercial code $S4$ shell element (fully integrated, 4 nodes). All analysis will be linear, Gauss quadrature will always be of 2×2, also isotropic material is always considered.

7.1 Clamped squared plate

The clamped square plate is an interesting benchmark problem with the objective of evaluating the behaviour of the displacement field of a plate submitted to a constant pressure over the plane with all sides clamped, this means that the edge nodes don't allow nor displacement nor rotation, this boundary condition is applied by forcing the displacement and the rotation on the nodes to be null. The approach to this problem can be simplified with symmetry boundaries given that the plate has two symmetry planes, hence only one quarter of the plane has to be represented. In this work, the plate representation used for IGA was not simplified, only for FEM.

The square plate has its sides parallel to the x and y directions withstands a pressure of P = 1 Pa, the side length assumes the value of L = 1 m, the elasticity modulus is E = 1.0920×10^{12} Pa, the Poisson coefficient is $\nu = 0.3$ and the plate thickness is $t = 0.001$ m.

The obtained out-of-plane displacement w can be normalized using the following expression,

$$w_{norm} = \frac{wD}{\mathrm{PL}^4}, \tag{7.1}$$

where L is the side length, P is the pressure applied on the plate, D is the constitutive matrix for thin plates. This last variable is obtained as [39],

$$D = \frac{\mathrm{E}t^3}{12(1-\nu^2)} \tag{7.2}$$

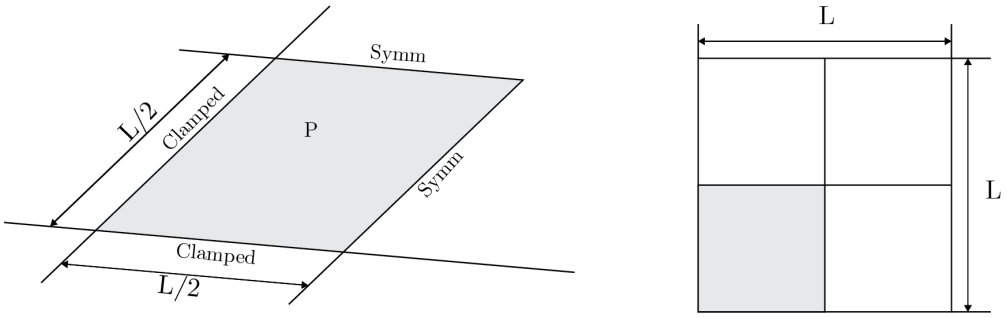


Figure 7.1: Clamped square plate representation.

Taking $w = 1.265 \times 10^{-5}$ m as the reference value. Figure 7.2 represents the convergence results for each one of the mentioned elements to the reference value. Figure 7.3 shows the relative error from the reference value in percentage. The convergence results relative to each element are the object of study. The number of degrees of freedom varies from a minimum of 36 to a maximum of 4761 d.o.f.'s. The number of d.o.f.'s for each element was chosen to be as close as possible to the other elements with a restriction, that the mesh has to have a middle knot where the displacement value is bigger. The results used to build the Figures 7.2 and 7.3 can be seen in the Tables B.1 and B.2, respectively.

Abaqus $S4$ element convergence is shown in the figure 7.2 as the dash-dot line with asterisk points. With increasing mesh refinement, its results convergence reach a relative error of around 0.1 %. The implemented MCZ thin plate element with three d.o.f.'s, is represented as the dashed line with cross points. With increasing mesh refinement, its results converged to a relative error of 0.1 %, although was very precise, it had a slightly slower convergence than *Abaqus* $S4$ element. Being both, thin plate and *S4* elements, processed by finite element method, it is important to compare their convergence rate. The fact that $S4$ element is a shell element with 3 points of integration through is thickness, gives more accuracy in comparison to a thin plate element, that only has integration points through the middle plane. On the other side, having less integration points makes this element faster to process.

The implemented NURBS-based thin plate element, was studied with different order meshes already announced. Represented in the Figure 7.2 with thick lines with a upward-triangle, a diamond and a square represent the $k = 3$, $k = 4$ and $k = 5$ orders, respectively. For NURBS-based thin plate elements the results converged to a relative error of close to 0.00 %. Considering the convergence rate, the higher orders $k = 4$ and $k = 5$ had a meaningful better rate than the order $k = 3$ seen in Figure 7.2. Although it was expected a better convergence rate for the higher order, it was order $k = 4$ that had the best convergence, even though it is a difference that can be despised given that, this is an approximation method and the relative error difference is bellow 1 % that can be seen in zoomed curve in Figure 7.3, this, however, can be justified by the utilization of the 2×2 Gauss integration points, since a $k + 1 \times l + 1$ integration is recommended.

On the overall of this benchmark, IGA NURBS-based thin plate element showed

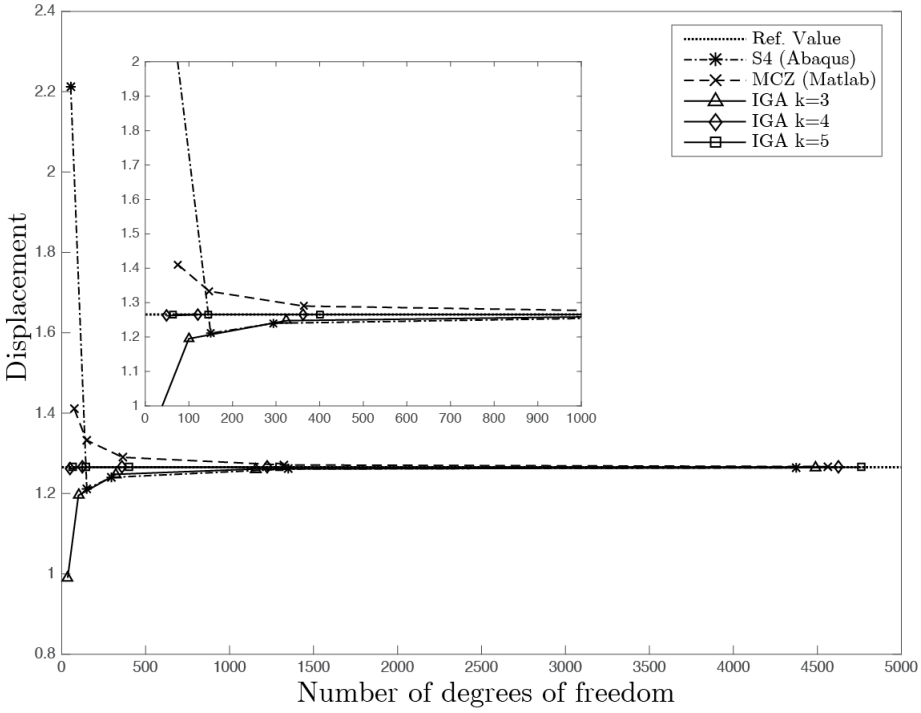


Figure 7.2: Convergence behaviour for each element when increasing the number of Degrees of Freedom.

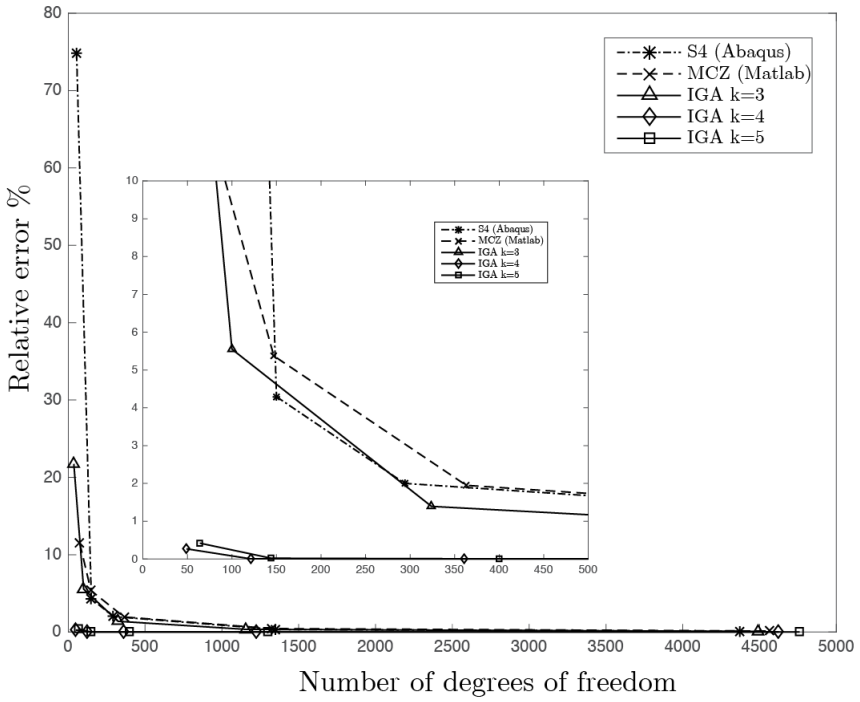


Figure 7.3: Relative error % from reference value.

the fastest convergence rate for higher orders and its results were the most accurate, on the other hand, the MCZ element showed the slowest convergence rate and, although precise, the results converged to the highest relative error.

Considering that IGA elements have more integration points hence they have fewer d.o.f.'s per element. The tables bellow shows the difference in precision when the elements are compared by integration points.

Table 7.1: Results from fixed square plate subject to uniform pressure.

Nº elements	Ref Value	Abaqus $S4$	MCZ (Matlab)	IGA ($p = 3$)	IGA ($p = 4$)	IGA ($p = 5$)
16(4x4)		1,2113	1,4105	0,9903	1,2618	1,2600
64(8x8)		1,2507	1,3042	1,1951	1,2652	1,2656
256(16x16)	1,2653	1,2615	1,2752	1,2477	1,2653	1,2654
1024(32x32)		1,2644	1,2677	1,2609	1,2653	1,2653
4096(64x64)		1,2650	1,2659	1,2642	1,2653	1,2653

Table 7.2: Relative error % from reference value.

Nº elements	Abaqus $S4$	MCZ (Matlab)	IGA ($p = 3$)	IGA ($p = 4$)	IGA ($p = 5$)
16(4x4)	4,27	11,48	21,73	0,28	0.42
64(8x8)	1,15	3,07	5,55	0,01	0,02
256(16x16)	0,30	0,78	1,39	0,00	0,01
1024(32x32)	0,07	0,19	0,35	0,00	0,00
4096(64x64)	0,02	0,05	0,09	0,00	0,00

7.2 Simply-Supported Square Plate

Simply supported square plate is another important benchmark in the study of the displacement field behaviour for the proposed thin plate elements. With, again, the main goal of evaluating the behaviour of the displacement over a plane, with the difference that as seen in figure, all its sides are simply supported, meaning that the boundary condition on the edge node is only null for displacement, given this, a bigger displacement of the middle node is expected in comparison to a clamped plate. The approach to this benchmark can also be simplified by studying only one quarter of the plate.

The square plate, has its sides parallel to the x and y directions, holds a uniform pressure all over its plane of P = 1 Pa, the square side length is $L = 1$ m, the elasticity modulus is E = 1.0920×10^{12} PA, the assumed Poisson coefficient is $\nu = 0.30$ and the plate thickness is $t = 0.001$ m.

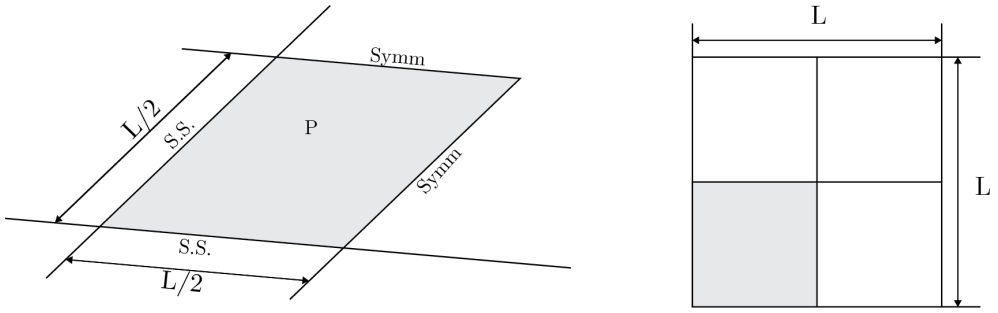


Figure 7.4: Simply supported square plate representation.

The out-of-plane displacement w can be normalized using the expression 7.1. Considering the reference value as $w = 4.0624 \times 10^{-5}$ m represented in the figure 7.5 as the straight continuous line. Figure 7.5 illustrates the convergence of each element in study to the reference value. Figure 7.6 shows the relative error in percentage from the reference value as the number of d.o.f.'s is increased, as the clamped case, the number of degrees of freedom varies from 36 to a maximum of 4761 d.o.f.'s.The results used to build the Figures 7.5 and 7.6 can be seen in the Tables B.3 and B.4, respectively.

Abaqus S4 element results, converged to a relative error of 0.03 %, for a maximum of 4374 d.o.f.'s the resulting deflection is slightly above of the reference value as seen in Figure 7.5. MCZ thin plate finite element converged to a similar relative error as Abaqus S4, although it showed a better convergence rate.

IGA NURBS-based thin plate element is represented in the Figure 7.5 represented with dashed line and a upwards triangle a diamond and a square for $k = 3$, $k = 4$ and $k = 5$, respectively. Considering the results of convergence for each order, $k = 3$ has a slower convergence rate than the higher orders but from around the 500 d.o.f.'s the result equals the reference value showing high accuracy. The higher orders $k = 4$ and $k = 5$, are accurate since the first analysis with lower mesh definitions.

On the overall higher order NURBS-based thin plate elements showed better convergence rates, and better accuracy for lower mesh definitions. Although all elements

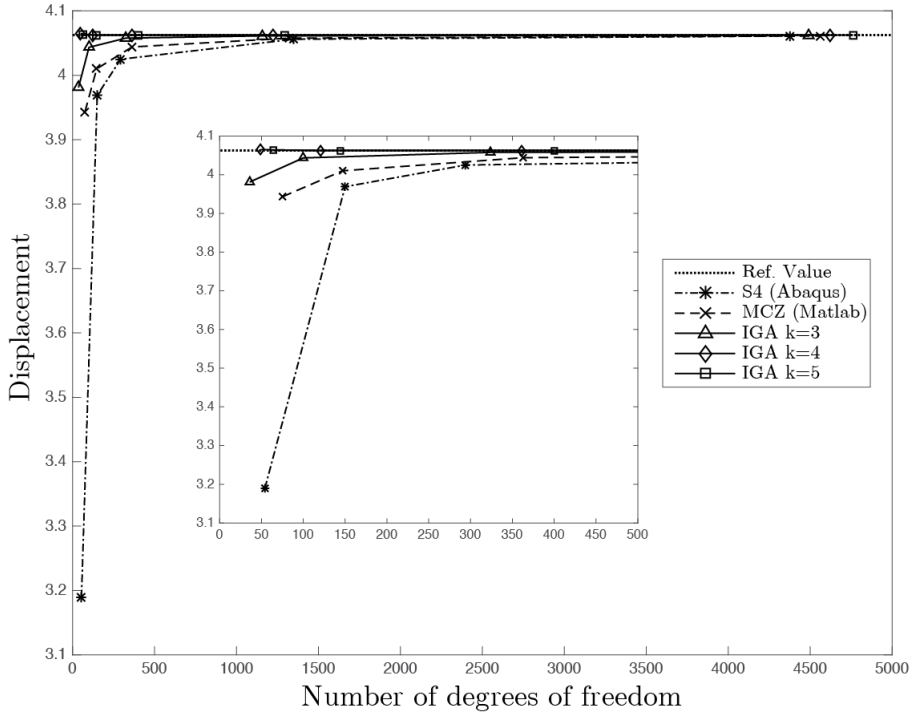


Figure 7.5: Convergence behaviour for each element when increasing the number of d.o.f.'s.

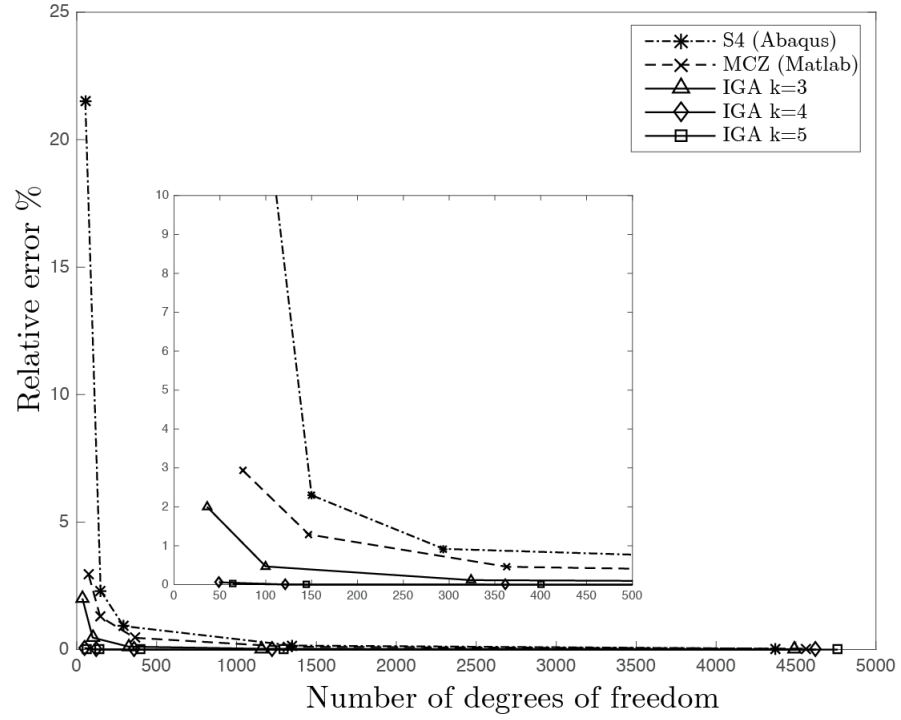


Figure 7.6: Relative error when increasing d.o.f.'s.

showed high accuracy for the most refined meshes with around 4500 d.o.f.'s.

Considering that IGA elements have more integration points hence they have fewer d.o.f.'s per element. The tables bellow shows the difference in precision when the elements are compared by integration points.

Table 7.3: Convergence results for a simply supported plate.

Nº elements	Ref Value	*Abaqus S4*	MCZ (Matlab)	IGA ($p = 3$)	IGA ($p = 4$)	IGA ($p = 5$)
16(4x4)		4,1120	3,9431	3,9812	4,0647	4,0631
64(8x8)		4,0747	4,0332	4,0434	4,0625	4,0624
256(16x16)	4,0624	4,0655	4,0551	4,0577	4,0624	4,0623
1024(32x32)		4,0631	4,0605	4,0612	4,0624	4,0624
4096(64x64)		4,0625	4,0619	4,0621	4,0624	4,0624

Table 7.4: Convergence Error % from reference value.

Nº elements	*Abaqus S4*	MCZ (Matlab)	IGA ($p = 3$)	IGA ($p = 4$)	IGA ($p = 5$)
16(4x4)	1,22	11,48	2,93	0,06	0,02
64(8x8)	0,30	3,07	0,72	0,00	0,00
256(16x16)	0,08	0,78	0,18	0,00	0,00
1024(32x32)	0,02	0,19	0,04	0,00	0,00
4096(64x64)	0,00	0,05	0,01	0,00	0,00

7.3 Morley's 30° Skew Plate

Morley's skew plate is a classical benchmark to evaluate the elements with a distorted mesh. Originally introduced by Morley [40], who produced a series of analytical solutions to the boundary value problem. The edges are all simply supported. The skew is defined, as seen in the figure, by a 30° angle. Geometrically all sides have the same length with a length-thickness ratio of $L/t = 100$ where $L = 100$ and $t = 1$. The material properties assume a Young modulus of $E = 10^5$ and a Poisson coefficient of $\nu = 0.3$. Kirchhoff solution 4.445 obtained by Morley is used often in thin plate literature and it is presented in the Figure 7.8 as the thin constant dotted line. This case presents a 0.01 thickness to length ratio, according to [41] the shear deformations cannot be neglected. Presented in the same figure as the thick dotted line with the constant value of 4.640 is the correct value according to [41]. The relative error presented is calculated using the normalized value of the maximum out-of-plane displacement obtained by the equation 7.1 in relation to the reference value.

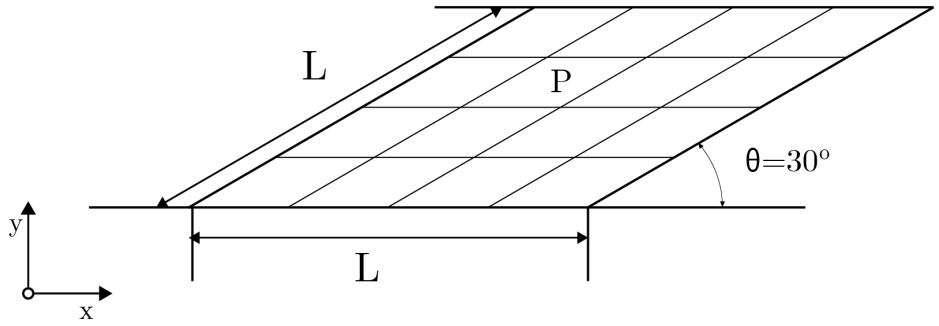


Figure 7.7: Morley skew plate representation.

Before going through the results convergence analysis, it is important to note that typically, the usage of a MCZ plate element is recommended with a regular rectangular $a \times b$ element mesh. As this benchmark needs the use of a distorted mesh, for sake of improving accuracy, going back to the definition of an MCZ thin plate element represented in the Figure 3.5 on Section 3.8, the length for the distorted side of the element is re-defined as follows,

$$2 \times b = L_e/\cos(30) \tag{7.3}$$

on the other hand, the sides on the x direction will continue to be defined by $2 \times a = L$.

Figure 7.8 represents the convergence curves of in study elements to both Morley and Andelfinger, Ramm reference values. As for Figure 7.9 represents the relative error curves of each element to Andelfinger and Ramm reference value.The results used to build the Figures 7.8 and 7.9 can be seen in the Tables B.5 and B.6, respectively.

Abaqus *S4* element is presented in the Figure 7.8 as the dot-dashed line with asterisk points, converged to a relative error of 4 %. Whereas for MCZ thin plate element represented with the dot-dashed line with cross points, the results tend to an relative error of around 8 %. Considering that the relative error from around 500 to the maximum

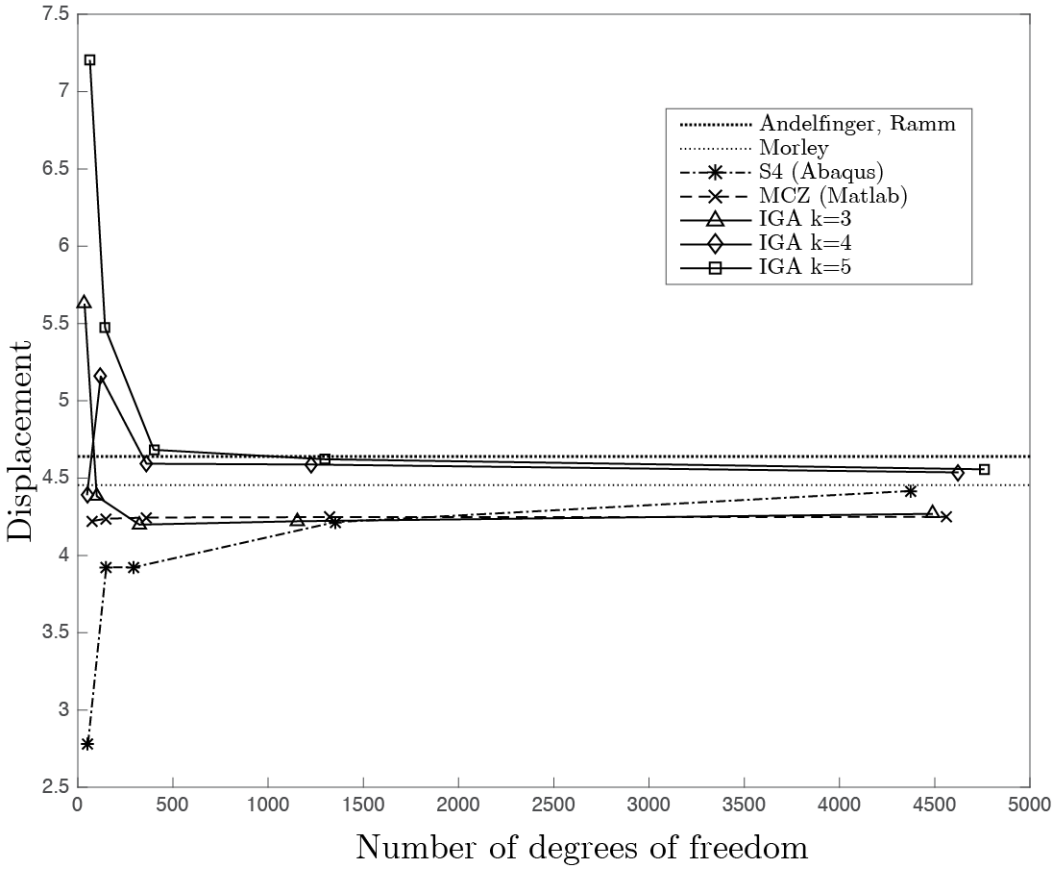


Figure 7.8: Convergence behaviour for each element when increasing the number of degrees of freedom.

d.o.f.'s is close, it is possible to conclude that for more refined meshes the results will be around 4.25. MCZ thin-plate element does not behave well using distorted elements. Also it neglects the contribution of shear stresses and that contributes for a bigger error.

NURBS-based thin plate element for a order of $k = 3$ showed a convergence similar to MCZ thin plate element, although it showed a oscillatory result convergence, the reached value of relative error was 8.00 %. Higher order elements $k = 4$ and $k = 5$ show very different convergence results also with oscillatory results, while $k = 4$ results on a convergence to a relative error of around 2 % the best accuracy mesh was with around 500 d.o.f.'s with a relative error of approximately 1.00 %. Order $k = 5$ present the biggest error of around 55 %, although its results converged to approximately 2 % error having the best accuracy with a number of 1250 d.o.f.'s resulting in a 0.39 % relative error. Again, it is important to acknowledge that the 2×2 Gaussian quadrature is not the recommended to an high order elements, which reflects, meaningfully on the bigger element meshes.

On the overall higher order NURBS-based thin plate elements showed better accuracy with refined meshes and better convergence rates, however $k = 3$ order had a very different convergence results similar to MCZ thin plate element. Abaqus $S4$ element behaviour was not what was expected, hence it is a shell element and it does not neglect

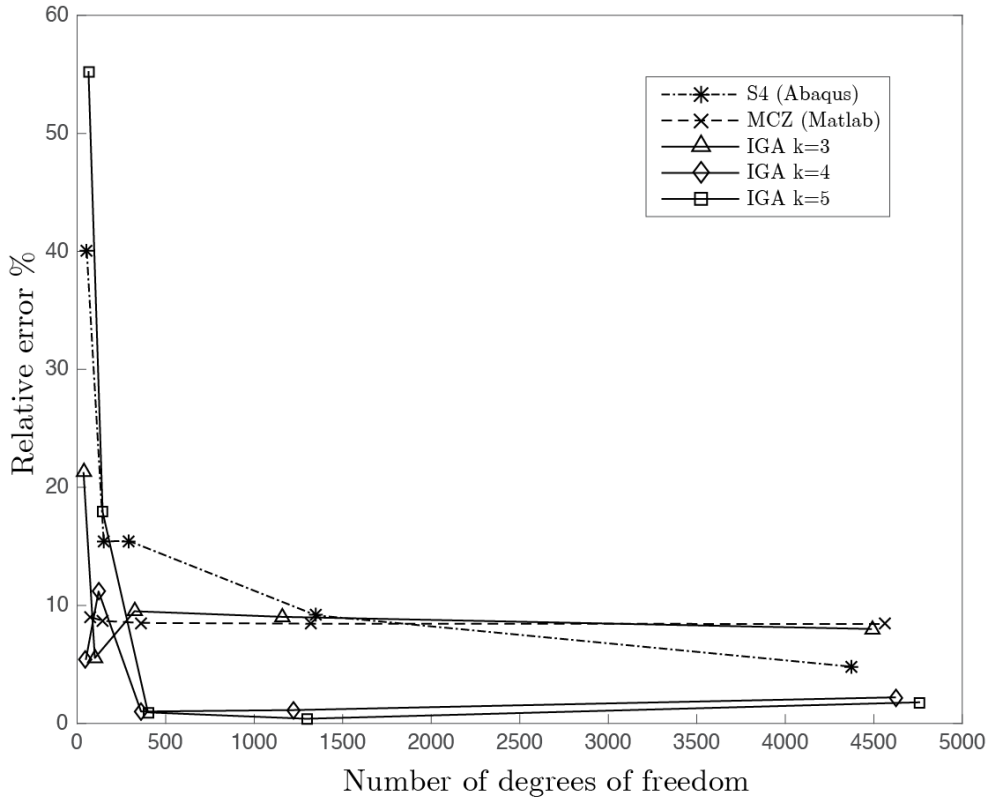


Figure 7.9: Relative error in % for Morley plate, with reference value presented in [41]

shear stress, it was expected this to be the most accurate element. Although, it is important to comprehend that being a 6 d.o.f. per node, the mesh for 4374 d.o.f.'s is not refined enough to analyse the complete convergence.

For better comprehension of the convergence, the Tables 7.5 and 7.6 give a comparison of convergence by the number of elements, which is merely representative, but it can be noted that with 4096 elements (24576 d.o.f.'s) Abaqus *S4* element starts to converge closer to the reference value.

Table 7.5: Convergence results for Morley's Plate.

Nº elements	Ref. Value	Abaqus $S4$	MCZ (Matlab)	IGA ($k=3$)	IGA ($k=4$)	IGA ($k=5$)
16(4x4)		3,924	4,222	5,628	4,390	7,204
64(8x8)		3,985	4,242	4,383	5,159	5,472
256(16x16)	4.640	4,262	4,248	4,199	4,593	4,683
1024(32x32)		4,445	4,249	4,221	4,588	4,622
4096(64x64)		4,553	4,249	4,269	4,537	4,557

Table 7.6: Convergence relative error % compared to Andelfinger and Ramm reference value.

Nº elements	Abaqus	MCZ (Matlab)	IGA ($k = 3$)	IGA ($k = 4$)	IGA ($k = 5$)
16(4x4)	15,43	9,01	21,29	5,39	55,26
64(8x8)	14,12	8,58	5,55	11,18	17,92
256(16x16)	8,15	8,45	9,50	1,00	0,93
1024(32x32)	4,20	8,43	9,03	1,13	0,39
4096(64x64)	1,87	8,43	8,00	2,21	1,79

Intentionally left blank.

Chapter 8

Concluding remarks and future works

Outlining the outcomes of this work, geometry description technologies were implemented, such as Bézier curves, B-splines and non-uniform rational B-splines. The comprehension of these formulations was fundamental to proceed with NURBS functions as basis for the analysis. A MCZ thin plate element was implemented and evaluated using the Finite Element Method. Then, a NURBS-based element, considering a somewhat similar formulation of MCZ, was implemented. The fact that the formulation of Isogeometric Analysis is rotation-free allows the analysis to be done on the real geometry and clears additional modification.

Considering the general behaviour of each element in numerical examples. MCZ thin-plate element showed a great performance when it came to the square plate benchmarks, but being a very restricted element, it showed lower flexibility for distorted meshes. Square plate results converged at great rates and with good accuracy to the reference value, whereas for the Morley plate (which trigger the need to a distorted mesh for analysis) the relative error grown to about 8%. Abaqus $S4$ shell element showed greater accuracy for the more refined meshes. In the Morley plate benchmark the convergence did not stabilized, and considering the Table 7.5 it can be seen that it is needed a more refined mesh to achieve stability in convergence. NURBS-based thin plate elements showed great convergence rate having its results to stabilize with the lower mesh definitions. All orders were highly accurate for square plate benchmarks having a relative error lower than 1%. As for Morley plate the converged results were not as accurate with respect to the reference value being the element of order $k = 5$ that presented the lower relative error of around 2%.

Exploring the smoothness of NURBS curves showed that Isogeometric Analysis can be very convenient when applied in thin plate analysis, since it can be difficult to achieve a C^1 continuity in conventional finite elements. The numerical results validated the benefits of using isogeometric discretization on the overall increasing the continuity between element with higher order NURBS lead to better convergence rates and accuracy. The higher order elements $k = 4$ and $k = 5$ shown highest accuracy and the fastest convergence.

Finite Element Method has captivated the market and is integrated in the industry for a long time, it is a very flexible method and even with its CAD-CAE communication

disadvantages it is very reliable. Whereas for Isogeometric Analysis the circumstances are different, being a method with about ten years of existence but that can bring a lot of advantages. Still being underdeveloped, it needs further investigation and build up to reach the market as it is still laying through mathematical algorithms. In future works the optimization of algorithms as well as a user interface must be developed. There are interesting areas that can be of reference for future works:

1. Non-linear analysis algorithms: in this work all analysis considered was linear elastic regime, it would be interesting to develop non-linear regime algorithms;
2. Development of IGA based Kirchhoff-Love shell elements: in this work only plate analysis was developed;
3. Development of IGA-based Mindlin-Reissner formulation for thick plate structures;
4. Development of T-spline algorithms in plate analysis;
5. Contact analysis.

Appendix A

Code Development

In this appendix file are represented the most important algorithms used in this thesis. Firstly with spline algorithms, used in geometry description for Isogeometric Analysis. Second there is MCZ thin plate element algorithm, where only the Stiffness Matrix routine with the element definition is shown. For Last the NURBS-based thin-plate algorithm where, also the stiffness matrix routine is presented.

A.1 Spline Code

This section is oriented as the Spline chapter beginning with the Bézier curves, being the most limited, through the B-splines until NURBS. For each geometry description technology it is presented it's curves and surfaces.

A.1.1 Bézier Curves

The following algorithm is used to plot a Bézier Curve. As announced in Section 4.2.1, the curve is defined with control points and the *nth*-degree Bernstein basis. The algorithm uses the function *allbernstein* from [2] to extract the Bernstein basis from incremented values between [0 1] based on the Equation 4.15, then multiplies these with the respective control points, given that it is created an coordinate array which is plotted.

Algorithm 1: Bézier Curve

1. Define the geometry properties;
   Control points - B
   Curve degree - n
2. Loop over the incremented values;
   $u = 0$: increment :1
   - *i.* Extract the Bernstein basis with the mentioned function;
     $J = allbernstein(u, n)$
   - *ii.* Multiply the basis with correspondent control points as in Equation 4.14;
   - *iii.* Add the point to the coordinate array - P

3 End Loop;

4 Plot the curve;

A.1.2 Bézier Surfaces

The algorithm for Bézier surfaces

Algorithm 2: Bézier Surface

1. Define the geometry properties;
Control points - B
Curve degree in each direction - n m

2. Loop over the incremented values in u direction ;
$u = 0$: increment :1

i. Loop over the incremented values in v direction ;
$v = 0$: increment :1

a. Extract the Bernstein basis for each direction;
$J = allbernstein(u, n)$
$K = allbernstein(v, m)$

b. Multiply the basis with correspondent control points as in Equation 4.21;

c. Add the point to the coordinate array - P

ii. End Loop;

3 End Loop;

4 Plot the surface;

A.1.3 B-spline Curves

The following Code is used to plot a B-spline Curve. Given the control points, the knot vector and the knot increment, there is one function that will find the knot index of the knot span *FindSpan* (adapted from [2]), and other function that with the knot index, calculates the basis through the Equations 4.32 and 4.33 *BasisFun* (adapted from [2]), then Equation 4.31 is applied to extract the curve points corresponding to the knot value.

Algorithm 3: B-spline Curve

1. Define the geometry properties;
Control points - B
Knot vector - U
Degree - q

2. Determine the number of control points - n

3. Specify the value increment of the knot span;

4. Loop over the increments until the maximum knot value
   $u = min(U)$: increment value : $max(U)$;

   i. Find the span of the knot increment;
      $i = FindSpan(n, u, q, U)$

   ii. Compute the basis function of the given span;
      $N = BasisFun(i, u, q, U)$

   iii. Multiply the basis with the corresponding control points 4.31;

   iv. Add the resulting points to the coordinate matrix - P;

5. End loop;

6. Plot the coordinate matrix to draw the B-spline curve.

A.1.4 B-spline Surfaces

For the B-spline surfaces the algorithm is similar from its curves but with two knot vectors,

Algorithm 4: B-spline Surface

1. Define the geometry properties;
   Control points - B
   Knot vector in each direction- U, V
   Degree of each knot vector- q, p

2. Determine the number of control points in each direction - n, m;

3. Specify the value increment of the knot span in both directions;

4. Loop over the increments until the maximum knot value in x direction;
   $u = min(U)$: increment value : $max(U)$

   i. Loop over the increments until the maximum knot value in y direction;
      $v = min(V)$: increment value : $max(V)$

      a. Find the span of the knot increment;
         $i = FindSpan(n, u, q, U)$
         $j = FindSpan(m, v, p, V)$

      b. Compute the basis function of the given span;
         $N = BasisFun(i, u, q, U)$
         $M = BasisFun(j, v, p, V)$

      c. Multiply the basis with the corresponding control points 4.37;

      d. Add the resulting points to the coordinate matrix - P;

   ii. End loop;

6. End loop;

7. Plot the coordinate matrix to draw the B-spline surface.

A.1.5 NURBS Curves

The following algorithm reproduces NURBS curves. The implementation follows the same steps of a B-spline curve, whereas the difference stands on the weights as announced on section 4.4.1.

Algorithm 5: Non-Rational Uniform B-spline Curve

1. Define the geometry properties;
   Control points - B
   Knot vector - U
   Degree - q
   Weights - w
2. Determine the number of control points - n
3. Multiply the weights with the control points vector to create the weighted control points vector;
   $B_w = B \times w$
4. Specify the value increment of the knot span;
5. Loop over the increments until the maximum knot value;
   $u = min(U)$: increment value : $max(U)$
   i. Find the span of the knot increment;
      $i = FindSpan(n, u, q, U)$
   ii. Compute the basis function of the given span;
      $N = BasisFun(i, u, q, U)$
   iii. Multiply the basis with the corresponding weighted control points;
      $C_w = N \times Bw$
   v. Divide the resulting four-dimensional curve C_w by the weight to obtain the three-dimensional point;
      $C = C_w/w$
   v. Add the resulting points to the coordinate matrix - P;
6. End loop;
7. Plot the coordinate matrix to draw the NURBS curve.

A.1.6 NURBS Surfaces

NURBS surfaces algorithm is very similar to its curves applying just another direction.

Algorithm 6: Non-Rational Uniform B-spline Surface

1. Define the geometry properties;

Control points - B
Knot vectors - U, V
Degree - q, p
Weights - w

2. Determine the number of control points in each direction - n, m;

3. Create the weighted control points vector;
$B_w = [B_x \times w \;\; B_y \times w \;\; B_z \times w \;\; w]$

4. Specify the value increment of the knot span;

5. Loop over the increments until the maximum knot value;
$u = min(U)$: increment value : $max(U)$

   i. Loop over the increments until the maximum knot value;
   $v = min(V)$: increment value : $max(V)$

      a. Find the span of the knot increment;
      $i = FindSpan(n, u, q, U)$
      $j = FindSpan(m, v, p, V)$

      b. Compute the basis function of the given span;
      $N = BasisFun(i, u, q, U)$
      $M = BasisFun(j, v, p, V)$

      c. Multiply the basis with the corresponding weighted control points vector;
      $Cw = N \times M \times Bw$;

      c. Divide the weighted control points vector by the weights;
      $C = C_w/w$

      d. Add the resulting points to the coordinate matrix - P;

6. End loop;

7. Plot the coordinate matrix to draw the NURBS Surface.

A.2 MCZ Thin-Plate Algorithm

The bellow announced algorithm is used to obtain the global stiffness matrix, having the geometry defined. The code runs trough all the elements that define the geometry and assemble them into the global stiffness matrix **K**. In each loop the element index is used to extract the nodes that define its boundaries, then it is determined the respective degrees of freedom of each node. Given that, a subroutine is put to run in which the element stiffness matrix **ke** and element force vector **fe** are calculated.

Algorithm 7: Global stiffness matrix

1. Define the plate geometry;
   Length- L
   Thickness- T
2. Specify the plate mesh connectivity;
3. Designate the material properties;
   Young modulus - E
   Poisson coefficient - ν
   Constitutive matrix - D
4. Define Gauss integration 2×2 - Q;
5. Input the plate pressure - P;
6. Loop through the elements;
   e=1:numelements
   i. Extract the boundary nodes;
      $i = element(e, :)$
   ii. Specify the corresponding degrees of freedom of each node;
   ii. Determine the element stiffness matrix and force vector;
      $[ke, fe] = MCZ_Plate(Coord, D, P, Q)$
   iii. Add the elementary contribution to the global stiffness matrix;
      $\mathbf{K} = \mathbf{K} + ke$
   iv. Add the elementary contribution to the global force vector;
      $\mathbf{F} = \mathbf{F} + fe$
7. End Loop.

Algorithm 8: Element stiffness matrix **ke** and force vector **fe**

1. Enable function $[\mathbf{ke}, \mathbf{fe}] = MCZ_Plate(Coord, D, P, Q)$
2. Define sides a and b;
   $a = max((coord(:, 1)) - min(coord(:, 1)))/2$
   $b = max((coord(:, 2)) - min(coord(:, 2)))/2$

3. Define element force vector;
$\mathbf{fe} = a \times b \times P/3 \times [3; b; -a; 3; b; a; 3; -b; a; 3; -b; -a]$

4. Loop through integration points;
$n = 1 : size(Q, 1)$

i. Computation of the bending strain matrix B_i

ii. Add nodal contributions to element stiffness matrix;
$\mathbf{ke} = ke + B_i^{\mathrm{T}} * D * B_i * a * b$

5. End loop.

A.3 NURBS-based MCZ thin-Plate Algorithm

This algorithm is used to obtain the global stiffness matrix **K**, as in FEM MCZ Thin-Plate, the geometry is defined first but with NURBS surface, after that all material properties are specified. As presented in chapter 5 the code also run through the defined elements, gauss quadrature is defined and the integration is made in a loop to assembly all node contributions. It is possible to acknowledge the differences as here the element is bounded by $k \times l$ nodes, the shape functions are the basis functions of the NURBS curve and there are two new reference spaces.

Algorithm 9: Global stiffness matrix **K** and global force vector **F**

1. Input NURBS characteristics;
   Control points - $ControlPts$
   Weights - $h_{i,j}$
   Order in each direction - k, l
   Knot vector in each direction - $KnotU$, $KnotV$
2. Build element connectivity matrix;
   $elements = [noElements, k \times l]$
   Index space - $Index = [unique(KnotU) \times unique(KnotV)]$
   Parent space - $\Omega_u = [\xi_i, \xi_{i+1}]$ and $\Omega_v = [\eta_j, \eta_{j+1}]$
3. Specify material properties;
   Young modulus - E
   Poisson coefficient - ν
   Constitutive matrix - D
4. Define the distributed load - P
5. Define the Gaussian quadrature of 2×2 - Q;
6. Loop over the elements;
   e=1:noElements
   - *i.* Extract parent index;
     $IndexU = Index(e, 1)$
     $IndexV = Index(e, 2)$
   - *ii.* Extract parent range;
     $\xi_i = \Omega_u(indexU, :)$
     $\eta_j = \Omega_v(indexV, :)$
   - *iii.* Extract the element scatter and node index;
     $sctr = elements(e, :)$
     $pts = controlPts(sctr, :)$
   - *iv.* Loop over the integration points;
     $n = 1 : size(Q, 2)$
     - *a.* Integration point;
       $pt = Q(n, :)$

b. Determine the parametric points from the parent space;

c. Determine Jacobian of parent to parametric space - $J1$;

d. Compute the basis functions and first and second derivatives;
$[R, R_{\xi}, R_{\eta}, R_{\xi\xi}, R_{\eta\eta}, R_{\xi\eta}]$ = NurbsBasisDerivatives(pt, k, l, $KnotU$, $knotV$, $h_{i,j}$)

e. Determine the Jacobian with respect to physical coordinates for first and second derivatives;
$J2$ - first derivative
$J33$ and $J32$ - second derivative

f. Compute the second derivative with respect to physical space Equation 6.10;
R_{xx}, R_{yy}, R_{xy}

g. Determine the bending strain matrix - B;

h. Determine the global stiffness matrix
$\mathbf{K}(sctr, sctr) = \mathbf{K}(sctr, sctr) + B^{\mathrm{T}} \times C \times B \times J1 \times J2$;

i. Determine the global force vector;
$\mathbf{F}(sctr) = \mathbf{F}(sctr) + P \times R^{\mathrm{T}} \times J1 \times J2$;

v. End loop

7. End loop

This algorithm makes use of a function adapted from the [2] called *dersbasisfunction* also to extract te first and second derivatives of the NURBS basis curve, also the mentioned *FindSpan* and *BasisFun* were used.

Algorithm 10: Nurbs basis and *2nd* derivatives

1. Enable $[R, R_{\xi}, R_{\eta}, R_{\xi\xi}, R_{\eta\eta}, R_{\xi\eta}]$ = NurbsBasisDerivatives(pt, k, l, $KnotU$, $knotV$, $h_{i,j}$)

2. Number of non-zero intervals in the knot vector in u and v direction;
$nU = size(KnotU) - 1 - p - 1$
$nV = size(KnotV) - 1 - q - 1$

3. Determine the knot span in both directions;
$spanU = BasisFun(nU, pt(1), p, KnotU)$
$spanV = BasisFun(nV, pt(2), q, KnotV)$

4. Compute the basis function;
$N = BasisFun(spanU, pt(1), p, KnotU)$
$M = BasisFun(spanV, pt(2), q, KnotV)$

5. Compute the 1*st* and 2*nd* derivatives of the basis function;
$dersN = dersbasisfuns(spanU, p, nU, pt(1), KnotU)$
$dersM = dersbasisfuns(spanV, q, nV, pt(2), KnotV)$

6. Loop through the control net in u direction
i=1:p+1

i. Loop in trough the control net in the v direction;
j=1:q+1

a. Compute the weight of the indexed knot

b. Compute the Sum(u,v) function and it derivatives shown in Equations 4.68 to 4.72

ii. End loop

6. End loop

7. Determine the Basis, its 1st and 2nd derivative for each control point using the Equations 4.73 and 4.74

Appendix B

Tables

Table B.1: Clamped plate results.

Reference Value	Abaqus $S4$	d.o.f.'s	MCZ (Matlab)	d.o.f.'s	IGA $k=3$	d.o.f.'s	IGA $k=4$	d.o.f.'s	IGA $k=5$	d.o.f.'s
	2,212	54	1,411	75	0,990	36	1,262	49	1,260	64
	1,211	150	1,333	147	1,195	100	1,265	121	1,266	144
1,26532	1,240	294	1,290	363	1,247	324	1,265	361	1,265	400
	1,261	1350	1,271	1323	1,260	1156	1,265	1225	1,265	1296
	1,264	4374	1,267	4563	1,264	4489	1,265	4624	1,265	4761

Table B.2: Clamped plate: Relative error in % with respect to the reference value.

Abaqus $S4$	Dofs	MCZ (Matlab)	d.o.f.'s	IGA $k=3$	d.o.f.'s	IGA $k=4$	d.o.f.'s	IGA $k=5$	d.o.f.'s
74,862	54	11,542	75	21,739	36	0,237	49	0,395	64
4,269	150	5,375	147	5,534	100	0,000	121	0,079	144
1,976	294	1,976	363	1,423	324	0,000	361	0,000	400
0,316	1014	0,474	1323	0,395	1156	0,000	1225	0,000	1296
0,079	4374	0,158	4563	0,079	4489	0,000	4624	0,000	4761

Table B.3: Simply supported plate results

Reference Value	Abaqus $S4$	d.o.f.'s	MCZ (Matlab)	d.o.f.'s	IGA $k=3$	d.o.f.'s	IGA $k=4$	d.o.f.'s	IGA $k=5$	d.o.f.'s
4.062	3,189	54	3,943	75	3,981	36	4,065	49	4,063	64
	3,969	150	4,010	147	4,043	100	4,063	121	4,062	144
	4,025	294	4,044	363	4,058	324	4,062	361	4,062	400
	4,056	1350	4,058	1323	4,061	1156	4,062	1225	4,062	1296
	4,061	4374	4,061	4563	4,062	4489	4,062	4624	4,062	4761

Table B.4: Simply supported plate: Relative error in % with respect to the reference value

Abaqus $S4$	d.o.f.'s	MCZ (Matlab)	d.o.f.'s	IGA $k=3$	d.o.f.'s	IGA $k=4$	d.o.f.'s	IGA $k=5$	d.o.f.'s
21,492	54	2,930	75	1,994	36	0,074	49	0,025	64
2,290	150	1,280	147	0,468	100	0,025	121	0,000	144
0,911	294	0,450	363	0,098	324	0,000	361	0,000	400
0,148	1350	0,098	1323	0,025	1156	0,000	1225	0,000	1296
0,025	4374	0,025	4563	0,000	4489	0,000	4624	0,000	4761

Table B.5: Morley plate results

Morley	Andelfinger, U Ramm, E	Abaqus	d.o.f.'s	MCZ (Matlab)	d.o.f.'s	IGA $k=3$	d.o.f.'s	IGA $k=4$	d.o.f.'s	IGA $k=5$	d.o.f.'s
		2,780	54	4,222	75	5,628	36	4,390	49	7,204	64
		3,924	150	4,237	147	4,383	100	5,159	121	5,472	144
4,455	4,640	3,923	294	4,245	363	4,199	324	4,593	361	4,683	400
		4,215	1350	4,248	1323	4,221	1156	4,588	1225	4,622	1296
		4,417	4374	4,249	4563	4,269	4489	4,537	4624	4,557	4761

Table B.6: Morley plate: Relative error in % with respect to the reference value of Andelfinger and Ramm.

Abaqus	Dofs	MCZ (Matlab)	d.o.f.'s	IGA $k=3$	d.o.f.'s	IGA $k=4$	d.o.f.'s	IGA $k=5$	d.o.f.'s
40,086	54	9,009	75	21,293	36	5,389	49	55,263	64
15,431	150	8,685	147	5,545	100	11,183	121	17,923	144
15,453	294	8,513	363	9,505	324	1,004	361	0,933	400
9,159	1350	8,448	1323	9,031	1156	1,129	1225	0,393	1296
4,806	4374	8,427	4563	8,000	4489	2,210	4624	1,789	4761

Bibliography

[1] T. J. R. Hughes, J. A. Cottrell, and Y. Bazilevs. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement, 2005.

[2] L. Piegl and W. Tiller. The NURBS Book, 1996.

[3] D. F. Rogers. *An introduction to NURBS: with historical perspective*. Elsevier, 2001.

[4] P. Bézier. Définition numérique des courbes et surfaces I. *Automatisme*, 11(12):625–632, 1966.

[5] G. Kirchhoff. Uber das Gleichgewicht und die Bewegung einer elastischen Scheibe. *Journal fur die Reine und Angewandte Mathematik*, 40:51 – 88, 1850.

[6] A. Love. The small free vibrations and deformation of a thin elastic shell. *Philosophical Transactions of the Royal Society of London. A*, 179:491–546, 1888.

[7] L. da Veiga, A. Buffa, C. Lovadina, M. Martinelli, and G. Sangalli. An isogeometric method for the Reissner-Mindlin plate bending problem. *Computer Methods in Applied Mechanics and Engineering*, 209:45–53, 2012.

[8] D.J. Benson, Y. Bazilevs, M. Hsu, and T.J.R. Hughes. Isogeometric shell analysis: The Reissner–Mindlin shell. *Computer Methods in Applied Mechanics and Engineering*, 199(5):276–289, 2010.

[9] D.J. Benson, Y. Bazilevs, M.-C. Hsu, and T.J.R. Hughes. A large deformation, rotation-free, isogeometric shell. *Computer Methods in Applied Mechanics and Engineering*, 200(13):1367–1378, 2011.

[10] D.J. Benson, S. Hartmann, Y. Bazilevs, M.-C. Hsu, and T.J.R. Hughes. Blended isogeometric shells. *Computer Methods in Applied Mechanics and Engineering*, 255:133–146, 2013.

[11] J. Kiendl, K.-U. Bletzinger, J. Linhard, and R. W{\"u}chner. No TitleIsogeometric shell analysis with Kirchhoff-Love elements. *Computer Methods in Applied Mechanics and Engineering*, 49:3902–3914, 2009.

[12] R. Echter, B. Oesterle, and M. Bischoff. A hierarchic family of isogeometric shell finite elements. *Computer Methods in Applied Mechanics and Engineering*, 254:170–180, 2013.

[13] T.-K. Uhm and S.-K. Youn. T-spline finite element method for the analysis of shell structures. *International Journal for Numerical Methods in Engineering*, 80(4):507–536, 2009.

[14] J. M. Kiendl. Isogeometric Analysis and Shape Optimal Design of Shell Structures. page 140, 2011.

[15] J. Lu. Isogeometric elements of smooth boundary. *Computer methods in applied mechanics and engineering*, 198(30):2391–2402, 2009.

[16] J. Lu and X. Zhou. Isogeometric model of continuum rod. *Computer Methods in Applied Mechanics and Engineering*, 200(1):233–241, 2011.

[17] R. J. Melosh. Structural analysis of solids. *Journal of the Structural Division*, 89(4):205–223, 1963.

[18] E. Oñate. *Structural Analysis with the Finite Element Method*, volume 1. Springer Science & Business Media, 2009.

[19] M. Augusta, A. Amaro, L. Roseiro, J. Cirne, and R. Leal. *Engineering Computation of Structures : The Finite Element Method.* Springer Science & Business Media, 2015.

[20] F. Teixeira-Dias, J. P. da Cruz, R. F. Valente, and R. A. de Sousa. *Método dos Elementos Finitos-Técnicas de Simulaçao Numérica em Engenharia.* ETEP, 2ª edition.

[21] E. Reissner. The effect of transverse shear deformation on the bending of elastic plates. *Journal of Applied Mechanics (ASME)*, 12:69–77, 1945.

[22] R. D. Mindlin. Influence of Rotatory Inertia and Shear on Flexural Motions of Isotropic, Elastic Plates. *Journal of Applied Mechanics-Transactions of the ASME*, 18:31–38, 1951.

[23] E. Oñate. *Structural analysis with the finite element method. Linear statics: vol 2: beams, plates and shells.* Springer Science & Business Media, 2013.

[24] O.C. Zienkiewicz, R.L. Taylor, and J.Z. Zhu. *The Finite Element Method: Its Basis and Fundamentals.* 6 edition, 2005.

[25] O. C. Zienkiewicz and Y. K. Cheung. The finite element method for analysis of elastic isotropic and orthotropic slabs. *Proceedings of the Institution of Civil Engineers*, pages 471–488, 1964.

[26] O. C. Zienkiewicz and Y. K. Cheung. No Finite element procedures in the solution of plate and shell problems. *Stress Analysis, Chapter*, 8:120–144, 1965.

[27] J. Bloomenthal and C. Bajaj. *The Geometry of Implicit Surfaces.* 1997.

[28] I. Schoenberg. Contributions to the Problem of Approximation of Equidistant Data by Analytic Functions. *Quarterly of Applied Mathematics*, 4(2):112–141, 1946.

[29] M.G. Cox. The numerical evaluation of B-splines. *National Physical Laboratory DNAC 4*, 1971.

[30] C. de Boor. On calculating with B-splines. *Journal of Approximation Theory*, 6(1):50–62, 1972.

[31] R. Riesenfeld. *Applications of B-spline approximation to geometric problems of computer-aided design.* PhD thesis, Syracuse University, Syracuse, NY, 1972. Also available as University of Utah, 1973.

[32] W. Gordon and R. F. Riesenfeld. Bernstein-Bézier Methods for the Computer-Aided Design of Free-Form Curves and Surfaces. *Journal of the ACM*, 21(2):293–310, 1974.

[33] F. Auricchio, F. Calabrò, T. J. R. Hughes, A. Reali, and G. Sangalli. A simple algorithm for obtaining nearly optimal quadrature rules for NURBS-based isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 249-252:15–27, 2012.

[34] T. J. R. Hughes, A. Reali, and G. Sangalli. Efficient quadrature for NURBS-based isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 199(5-8):301–313, 2010.

[35] E. Cohen, T. Lyche, and R. Riesenfeld. Discrete B-splines and subdivision techniques in computer-aided geometric design and computer graphics. *Computer graphics and image processing*, 14(2):87–111, 1980.

[36] H. Prautzsch. A short proof of the Oslo algorithm. *Computer Aided Geometric Design*, 1(1):95–96, 1984.

[37] L. Piegl and W. Tiller. Software engineering approach to degree elevation of B-spline curves. *Computer-Aided Design*, 26(1):17–28, 1994.

[38] E. Cohen, T. Lyche, and L. Schumaker. Algorithms for degree-raising of splines. *ACM Transactions on Graphics (TOG)*, 4(3):171–181, 1985.

[39] L. Kui, G. Liu, and O.C. Zienkiewicz. A generalized displacement method for the finite element analysis of thin shells. *International journal for numerical methods in engineering*, 21(12):2145–2155, 1985.

[40] L. Morley. *Skew plates and structures.* Pergamon Press, distributed in the Western Hemisphere by Macmillan, New York, 1963.

[41] U. Andelfinger and E. Ramm. EAS-elements for two-dimensional, three-dimensional, plate and shell structures and their equivalence to HR-elements. *International Journal for Numerical Methods in Engineering*, 36(8):1311–1337, 1993.